Network Working Group
INTERNET-DRAFT
Intended Status: Informational

J. Bi
Tsinghua University
H. Rafiee
V. Choudhary
J. Strassner
Huawei
D. Romascanu
Avaya

Expires: November 19, 2015
May 19, 2015

Simplified Use of Policy Abstractions (SUPA) Gap Analysis
<draft-bi-supa-gap-analysis-03.txt>

Abstract

As operators struggle to optimize their network for different applications while maximizing network resources usage, there’s growing business pressure to minimize operational tasks and the deployment time of new services. New automation paradigms are meant to help reach these goals, including the optimization of network functions through application control. This control could be signaled directly by an application, through a proxy or orchestrated in a centralized manner. The purpose of SUPA is to develop a methodology by which network services can be managed using standardized policy rules. SUPA will focus in the first phase on inter-datacenter traffic management as part of the distributed data center use case, including the automated provisioning of site-to-site virtual private networks of various types.This memo analyses the current state of the art of the industries in IETF and outside IETF.

Status of this Memo

Table of Contents

1. Introduction

Network operators, including Internet Service Providers, Datacenters operators and others, are under constant pressure to optimize the usage of their installed network resources while maintaining high availability, complexity and deploying new services at an ever-increasing pace. The introduction of new paradigms aims at reducing these efforts, optimized network resource usage and minimize operational overhead. Such a new paradigm is the deployment of automated network configuration and optimization through the use of two complementary mechanisms that are software abstractions to simplify monitoring and control operations and the increase in programmatic control over the configuration and operation of such networks. Policy-based management can be used to combine these two mechanisms into an extensible framework.

Management applications would benefit from a view of the network that is adapted to their needs and from a policy framework that is efficient and simple to use. Several organizations have started working on protocols and models to be used between controllers and network devices, either physical ones or virtualized. This work started some years ago in a number of different organizations and has spawned a large amount of interest in the networking community. However the definition of interfaces between controllers and applications, the so-called "northbound" side, has seen a lot less progress during the same time. There’s a need for management applications to interface with controllers in a simple and elegant way. For this purpose, applications require a way to express their requirements in the form of simple policy statements applied to network elements. These network elements should be as simplified (abstracted) as possible for their manipulation by the application. The responsibility of providing an abstract and simple view adapted to each application need is the burden of the controller. The goal of the Simplified Use of Policy Abstractions (SUPA) group is to develop a methodology by which network services can be managed and automated by using a set of information policy model and how these model can map to YANG-based service and policy data models. It also focus on how to communicate these policy models. SUPA will focus in the first phase on inter- datacenter traffic management as part of the distributed data center use case, including the set of information models required to construct an extensible, policy-based framework. These information models will lead to a set of core YANG data models for a policy-based management framework to monitor and control network services.

2. Scope and target for SUPA

SUPA introduces the concepts of multi-level (multiple levels of abstraction) and multi-technology (e.g., IP, VPNs, MPLS) network abstractions to address the current separation between development and deployment operations. Multiple levels of abstraction enable

common concepts present in different technologies and implementations to be represented in a common manner. This facilitates using diverse components and technologies to implement a network service. The following standard generic YANG-based service and policy data models are within the scope of SUPA working group: o model of the physical and virtual network topology including the resources (e.g., data rate or latency of links) and operational parameters needed to support service deployment over the network topology. o model of the network service (e.g., VPNs) and the network resources required by the network service to be correctly deployed and executed on the physical and/or virtual topology. o model of policy rules for managing the network service and mapping services dynamically to the network topology and network resources. Using the above models, service specific policy data models will be derived from a generic policy model, ensuring that policies have a common structure and can be easily reused as managed objects.

3. Related work within the IETF

3.1. I2RS Working Group

They are not working on interconnection of data centers and considering multi-tenant environment where having a possibility that each tenant control (config, modify, etc.) its whole network that might be physically located on different data centers simply without even the need to involve in its complex communication processes. In other word, SUPA wants to serve a user a service and the interaction to a user is also important. This is not true for I2RS as it focuses on the processes and uses programmable synchronize interfaces to transfer states to and out of the internet routing systems. This is true that I2RS WG also uses the Yang model, however, the model introduced in [yang-i2rs] is so general and not only specific to use cases defined in I2RS. In other word, for I2RS, yang model can help a network controller to understand the topology of the whole network and compare it with what it has and update the topology as needed. Therefore, the general model introduced in [yang-i2rs] can also be used as a base for SUPA.

3.2. L3SM Working Group

This working group focuses on communication of operators and customers by allowing customers to configure the network elements via layer 3 VPN technology. The proposal is very specific about using layer 3 VPN technology via MPBGP. This group also wants to use Yang model to be able to configure network devices.

The differences of this group with SUPA is as followings:

- SUPA proposes a generic proposal for various VPN technologies like L2VPN, L3VPN and composite VPNs. Moreover, the proposed framework is

flexible enough to meet the requirement of any of the existing or upcoming VPN technologies.

- L3SM is more inclined towards MPLS/BGP VPN usecase but SUPA does not focus on a specific use case.

- L3SM focus only on configuration and has no provision for monitoring but SUPA provides service monitoring flexibility.

- L3SM charter did not explain anything about having a network controller and only focuses on device configuration via a L3VPN. In other word, customers might need to have different L3SM to configure different devices. While in SUPA, a management system would allow a customer to configure all or any selected devices concurrently via a network control.

The result of L3SM might be able to feed SUPA with their model to support policy information exchange in Layer 3 and SUPA might want to extend their model to use for SUPA-specific purposes.

3.3. ALTO Working Group

The ALTO working group defined an architecture for exposing topology information, more specifically the cost of paths through an infrastructure, as defined in [RFC7285]. ALTO services are able to provide network maps defined as groups of endpoints. Endpoints are providers-defined entities and can therefore represent any granularity of network, from the physical to groups of networks following similar paths or restrains. Although this model can represent different levels of abstraction at multiple granularities, it’s not clear if it could be adapted easily for other purposes than providing cost maps in the context of ALTO. The ALTO model is meant to be used outside of the trust domain of an ISP toward external clients.

3.4. TEAS Working Group

The Traffic Engineering Architecture and Signaling working group is responsible of MPLS-based Traffic Engineering, in other words the control of traffic flows in an MPLS network. It covers YANG models for a traffic engineering database, in coordination with other working groups (I2RS) providing YANG models for network topologies.

3.5. BESS Working Group

The BGP Enabled Services working groups aims at providing a protocol for the provisioning of L3VPN and L2VPN solutions based on BGP. This includes BGP-enabled solutions for datacenter networking and extensions to BGP-enabled solution to support Service Function Chaining. The working group is also chartered to work on on BGP

extensions to YANG models and data models for BGP-enabled services.

3.6. SFC Working Group

The Service Function Chaining (SFC) working group defines a mechanism where traffic is classified before going through an ordered set of services. The set of services is a definite and ordered group of instances defining a service function path. More than one instance may exist for each service in order to allow for load-balancing. A YANG definition for SFC is already proposed in [sfc-yang] and has been subject to an early implementation in Open Daylight. This interface and its model, as currently defined, is an abstraction limited to the scope of service chains.

3.7. NVO3 Working Group

The NVO3 group proposes a way to virtualize the network edge for datacenters in order to be able to move virtual instances without impacting their network configuration. This is realized through a centrally controlled overlay layer-3 network, as described in draft-lasserre-nvo3-framework. At first sight, there doesn’t seem to be an overlap between this work and what is being proposed in SUPA. This type of architecture could support a virtual tenant model similar to what is proposed in Open Daylight, but does not offer policing or new models for applications to use.

3.8. ACTN Proposed Working Group

The ACTN proposed work, as described in [actn] framework, has two main goals, the abstraction of multiple optical transport domains into a single controller offering a common abstract topology and the splitting of that topology into abstract client views, which are usually a fraction of the complete network. The ACTN work is therefore about unification of several physical controllers in a virtual one and also about the segmentation, isolation and sharing of network resources. The ACTN work is not explicitly about policies, but some level of policing is applied in the creation of a client view and the way it interacts with the virtual controller beneath. One point where overlap may exist with some of the work proposed in SUPA is in the definition of multiple levels of abstract topologies.

4. Related work outside the IETF

4.1. TM Forum

The TM Forum (a.k.a., the TeleManagement Forum) develops standards and best practices, research, and collaborative programs focused on digital business transformation. It consists of three major programs:

1) Agile Business and IT

2) Customer Centricity (experience)

3) Open Digital Ecosystem

Of these, the ZOOM (Zero-touch Orchestration, Operations, and Management) project, located in the Agile Business and IT project, is the main sub-project in this area that is of interest to SUPA.

Within ZOOM, the Foundational Studies project contains work on an information model and management architecture that are directly relevant to SUPA. The Information Model, Policy, and Security working groups are involved in this work.

The ZOOM information model updates the existing Shared Information and Data (SID) information model to add support for the management of physical and virtual infrastructure, event- and data-driven systems, policy management (architecture and model), metadata for describing and prescribing behavior that can support changes at runtime, and access control.

The policy information model defines event-condition-action (ECA), declarative (intent-based), utility function, and promise policies. The work in draft-strassner-supa-generic-policy-info-model-01 is based on this work. It currently extends the ECA model and provides additional detail not currently present in ZOOM; the next version of this draft will do the same for declarative policies.

There is currently no plan to use the utility function and promise policies.

Finally, it should be noted that the data model work planned for SUPA is not currently planned for the ZOOM project.

4.2. MEF

The MEF (originally named the Metro Ethernet Forum) develops architecture, service and management specifications related to Carrier Ethernet (CE). The CE architecture includes the definition of several interfaces specific to CE like the User Network Interface (UNI) and External Network Network Interface (ENNI). Specifications developed in this space include the definitions of CE services, CE service attributes, Ethernet Access Services, Class of Service, OAM and Management interfaces, Service Activation and Test.

The more recent vision of the MEF related to the future of networking is described as The Third Network and includes plans to develop Lifecycle Service Orchestration with APIs for existing network, NFV and SDN implementations enabling Agile, Assured and Orchestrated

Services. This stage of the MEF activity is now in early phases with focus on architectural work.

The MEF has developed a number of Information and Data Models, and has recently started a project that used YANG to model and manage the services covered by the MEF. Although the MEF has created quite rigorous definitions of these services, these are transport technology specific, and they do not include and rely on policies.

4.3. Open Daylight

Open Daylight network controller implements a number of models through its service abstraction Layer (MD-SAL) based on draft IETF Yang models. Few of the below mentioned Open Daylight projects provides policy abstraction and better flexibility to the user.

4.3.1. Network Intent Composition (NIC)

Network Intent Composition project aims at providing better flexibility and high-level interface for the specification of policies. The intents-based interface would provide a high level of abstraction easy to formulate by an application developer and completely detached from the underlying implementation details. By making intents portable and composable, the project aims at making intents a more scalable approach than existing interfaces.

4.3.2. Group Policy

The group-based policy project defines an application-centric policy model for Open Daylight that separates information about application connectivity requirements from information about the underlying details of the network infrastructure.

4.4. Open Networking Foundation

The ONF created a group responsible of defining northbound interfaces, but this hasn’t lead to the publication of standards in this area so far. A blog entry on the ONF web site showed an interest in using the principle of intents at ONF, but no details were provided on the status of this project. The membership of this group being closed in nature, the status of current draft proposals is not known.

4.5. OpenStack

OpenStack software controls large pools of compute, storage, and networking resources throughout a datacenter, managed through a dashboard or via the OpenStack API. OpenStack works with popular

enterprise and open source technologies making it ideal for heterogeneous infrastructure. Few of the below mentioned OpenStack projects provides policy abstraction and better flexibility to the user.

4.5.1. Group-Based Policies

The Group-Based Policies project for OpenStack Neutron is built around entities assembled in Endpoints Groups (EPG) that provide or consume Contracts. Such Contracts are hierarchical entities containing policy rules. A first version was released in January 2015, based on the Juno release. This type of approach is more relational than declarative, but could be used to describe a large amount of possible scenarios. It has the advantage of providing a relatively simple policy model that covers a large applicability. From an OpenStack point of view, the scope of GBP is limited to networking within the Neutron module.

4.5.2. Congress

The Congress project within OpenStack provides a way to formulate complex policies using the Datalog language, a derivate of Prolog. Datalog is entirely declarative and first-order logic, which gives it interesting properties, such as providing the same result no matter the order in which the statements are made. The language allows for the definition of types and for active enforcement or verification of the policies. Using Datalog also allows Congress to take advantage of the significant body of knowledge and experience relating to declarative languages and their implementation. The Congress policies aim at manipulating objects exposed by multiple OpenStack modules and is therefore larger in scope than network policying only. The only drawback of this approach lies in its potentially large computational complexity, which could limit its ability to react in real time fast events such as those relating to the network.

4.6. The NEMO Project

The NEMO project is a research activity aiming at defining a simple declarative framework for networking. The NEMO syntax is not based on an existing language and covers the basic elements for network manipulation such as nodes, links and flows. The NEMO project has been successfully demonstrated at IETF-91, along with a companion graphical user interface, and this work now being proposed as the base for a new group called Intent-Based NEMO (IBNEMO) within the IETF.

4.7. The Floodlight Project

The Floodlight is an openflow enabled SDN controller. it uses another

open source project called Indigo to support openflow and manage southbound devices. Indigo agent also supports abstraction layer to make it easy to integrate with physical and virtual switches. It supports configuration of abstraction layer so that it can configure openflow in hybrid mode.

4.8. The ONOS Project

The ONOS is a SDN controller. It supports abstraction for both southbound and northbound interfaces. This is because NFV used in ONOS can reduce CaPex because each service can be virtualized, but it increases OpEx as service providers now have to contend with increased management complexity due to the management and orchestration of a large numbers of VMs on commodity servers, the management of network function software on the VMs and how VMs must be interconnected based on subscriber’s service contract. This is why, ONOS uses Network Function as a Service (NFaaS) that not only virtualized the components and Virtual Network Functions (VNFs) but also introduces an abstractive unit for a collection of VMs and their interconnecting network(s). Being able to create and manipulate these units, rather than handling individual components, significantly simplifies operation. NFaaS manipulates these units in the enhanced form of a service.

5. Discussion

The ongoing projects outside of the IETF (see Section 4) demonstrate that there is a need to develop service level abstractions and policies that govern their implementation and mapping to the underlying network infrastructure. While different approaches are currently being prototyped, it is desirable from an operator’s perspective and of likely also of strategic importance from an IETF’s perspective to host work in this area within the IETF with a goal to drive progress towards a common standardized solution in this space. Generic policy driven service management is not directly worked on by existing IETF working groups. Several working groups provide technology specific mechanisms (TEAS, BESS, ACTN) that ideally can be leveraged by a generic policy driven service management solution. Other working groups provide key building blocks (e.g., the generic topology work recently chartered in the I2RS working group) or they look at specific aspects such as the chaining of data plane traffic manipulation functions (SFC) or the movement of virtual machines (NVO) or the export of typically aggregated topology information to distributed file sharing or streaming applications (ALTO).

6. Security Considerations

TBD

7. IANA Considerations

There is no IANA consideration

8. Acknowledgements

Jean-Francois Tremblay from Viagenie contributed to some significant portions of this text. James Huang, Oliver Huang, Will Liu, Yiyong Zha and Dacheng Zhang helped in providing valuable comments and text.

9. References

9.1. Normative References

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6241] Enns, R., Bjorklund, M., Schoenwaelder, J., and A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

[RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014

[yang-i2rs] Medved, J., Varga, R., Tkacik, T., Bahadur, N., Ananthakrishnan, H., "A Data Model for Network Topologies", https://tools.ietf.org/html/draft-ietf-i2rs-yang-network-topo-00, April 2015

[yang-l3sm] Litkowski, S., Shakir, R., Tomotaki, L., D'Souza, K., "YANG Data Model for L3VPN service delivery", https://tools.ietf.org/html/draft-l3vpn-service-yang-00, February 2015

Authors' Addresses

Jun Bi
Tsinghua University
Network Research Center, Tsinghua University
Beijing 100084
China
E-mail: junbi@cernet.edu.cn

Hosnieh Rafiee
Huawei Technologies Duesseldorf GmbH
Munich, Germany
Phone: +49 (0)162 204 74 58
E-mail: ietf@rozanak.com

Vikram Choudhary
Huawei Technologies
E-mail: vikram.choudhary@huawei.com

Dan Romascanu
Avaya
Atidim Technology Park, Bldg. #3
Tel Aviv 61581, Israel
Phone: +972-3-6458414
E-mail: dromasca@avaya.com

John Strassner
Futurewei Technologies
US R&D Center
2330 Central Expressway
Building A, office A2-2143
Santa Clara, California 95050
E-mail: john.sc.strassner@huawei.com

Network Working Group
Internet Draft
Intended status: Standard Track
Expires: November 2015

J. Bi
Tsinghua University
R. Tadepalli
Wipro Limited
M. Hayashi
KDDI R&D Labs. Inc.

May 7, 2015

DDC Service Policy YANG Data Model
draft-bi-supa-policy-model-02

Abstract

This document describes a YANG data model for traffic steering policies in Distributed Data Center (DDC) scenarios. The policy model is a specific data model for traffic steering using VPN technology. It helps the service management in Simplified Use of Policy Abstractions (SUPA) to model the policy (a set of constraints and rules) that defines how a VPN service is monitored by bandwidth and managed during its lifecycle. Two traffic steering applications have been provided such as traffic optimization with pass or bypass specific nodes or sites.

Status of this Memo

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

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

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

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

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

This Internet-Draft will expire on October 25, 2015.

Table of Contents

1. Introduction

In order to support the DDC service with VPN connection as well as new services, it brings new requirements for both network providers and service providers. Rapid uptake of new services requires dynamic service provisioning capabilities in the service management. This is achieved using policies that can be created by the operators once and the service management refers to these policies to infer how a given service needs to be provisioned considering the current state of the network.

In SUPA framework, network policy is a predefined rule or a set of rules that the service management use to map the service to the lower level network infrastructures. More specifically, based on different level of policy abstractions, there are a generic policy model on top and ECA policy, intent policy to handle service management. Note that, the generic policy is use case independent while the ECA policy has to be defined with objects from service model. In this way, an ECA policy defines:

An event or a set of events that trigger the evaluation of policy: This is the trigger for the service management application to evaluate if a policy needs to be applied. For example a user action to provision a new VPN service can be an event.

A set of conditions that need to be satisfied for the policy to be applicable: This enables service management to select the right policy by validating the conditions against the current network state.

A set of actions that should be triggered as part of the policy execution: This enables the service management to provision the service.

Meanwhile, DDC service which is mainly relied on VPN [RFC4110] needs policy based management and controlling capability from the service management systems to facilitate the service deployment both inter data centers and within data center.

This document introduces YANG [RFC6020] [RFC6021] data models for SUPA configuration. Such models can facilitate the standardization for the interface of SUPA, as they are compatible to a variety of protocols such as NETCONF [RFC6241] and [RESTCONF]. Please note that in the context of SUPA, the term "application" refers to an operational and management applications employed, and possibly implemented, by an operator. The policy model is based on the first example - DDC services.

With respect to the scope, defining an information model for the policy exchange between the policy manager and policy agent and a corresponding data model based on yang to support specific DDC service use case is initial goal of this document. The protocol specific aspects are deferred to respective implementations. Also certain foundational concepts of the model are intentionally left open to enable future extension. Here the traffic steering policy in DDC use case provides a concrete example for a specific network technology and service, as what constitutes a policy could itself vary depending on the context where it is used, e.g. there could

be tenant specific policies, site specific, network domain specific etc.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

3. Network Configuration Model Overview

Figure 1 illustrates the network configuration model which contains several modules for specific services such as VPN management. Basically, service model is to define the creation and configuration of the VPN service, while the policy model is more focused on the adjustment or optimization of the flow path during the lifecycle of the VPN service based on the predefined policy.

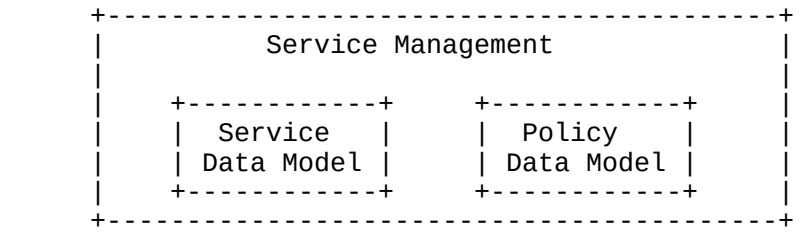


Figure 1: Overview of configuration model structure

4. Network Policy Configuration Modules

In this section, a policy model is defined with an application for traffic steering between data centers are provided to illustrate how to use the policy model. The policy model and policy configuration are based on a set of specific network services and the framework of SUPA [SUPA-framework]. On the other hand, the policy model should be working on the orchestration level which is above network element and below OSS level based on the YANG model classification in [draft-bogdanovic-netmod-yang-model classification-02]

4.1. Network Policy Model

A Policy Rule can be expressed in different ways and its content is separated from its representation. Therefore, this object takes different forms, depending on the level of abstraction desired. For example, an event-condition-action policy is expressed as a tuple of three statements. Within SUPA scope, the service management system takes the E-C-A fashioned policy model to handle the VPN management in the data center use case to improve bandwidth utilization and facilitate service deployment.

Event: a significant occurrence in time that triggers the evaluation of the condition of the policy rule

Condition: a set of tests that determine if the actions of the policy rule should be executed or not

Action: a set of operations that should be performed if the condition is true

Note that event, condition, and action can each be made up of Boolean clauses

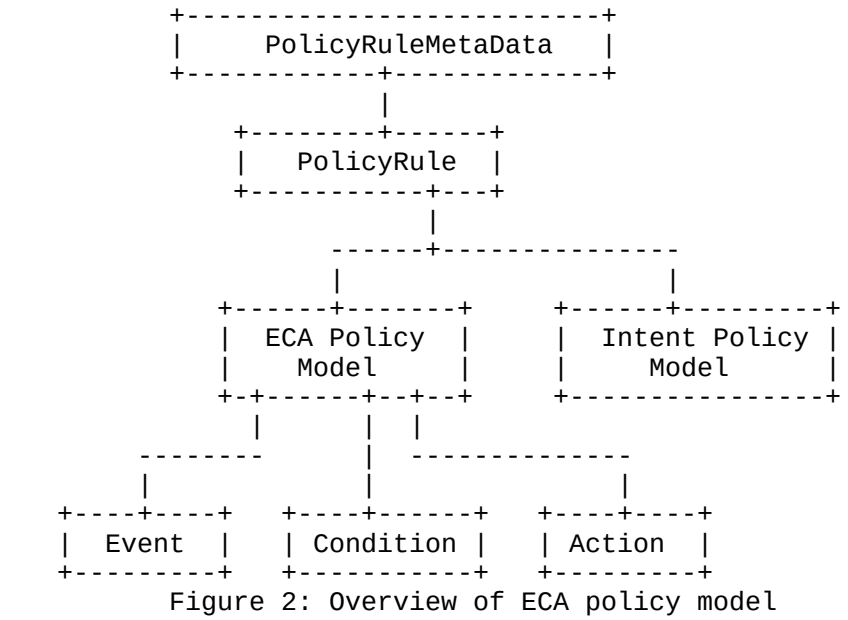


Figure 2: Overview of ECA policy model

```
module: ietf-supa-policy
   +--rw supa-policy
      +--rw supa-policy-name?              string
      +--rw supa-policy-type?              enumeration
      +--rw applicable-service-type?       enumeration
      +--rw supa-policy-priority?          uint8
      +--rw supa-policy-validity-period
      |  +--rw start?         yang:date-and-time
      |  +--rw end?           yang:date-and-time
      |  +--rw duration?      uint32
      |  +--rw periodicity?   enumeration
      +--rw supa-policy-metadata?          uint32
      +--rw supa-policy-atomic
      |  +--rw supa-ECA-policy-rule
      |     +--rw policy-rule-name?            string
      |     +--rw policy-rule-type?            enumeration
      |     +--rw policy-rule-deploy-status?   enumeration
      |     +--rw policy-rule-exec-status?     enumeration
      |     +--rw has-policy-events?           boolean
      |     +--rw has-policy-conditions?       boolean
      |     +--rw has-policy-actions?          boolean
      +--rw supa-policy-statement
         +--rw event-clause
         |  +--rw policy-variable?   string
         +--rw condition-clause
         |  +--rw policy-variable?   string
         |  +--rw policy-operator?   string
         |  +--rw policy-value?      string
         +--rw action-clause
            +--rw policy-action?   string
```

As shown in the YANG tree of the ECA policy model, it is basically following the generic policy model from [draft-strassner-supa-generic-policy-model-00] to define the policy model structure of the ECA policy.

The top level class "supa-policy": it has some basic attributes such as name and type. "applicable-service-type" is related the service which define the service being supported by the policy model. "supa-policy-priority" and "supa-policy-validity-period" define when and how often the policy will be executed, which is designed to handle some time based management policy. E.g. "perform LOAD BALANCING at 2am every Thursday" such policy will

introduce the timing issue. And finally, "supa-policy-atomic" and "supa-policy-statement" are two major components of the policy model.

"supa-policy-atomic" is a type of Policy Container which here contains the ECA model policy model. In addition to the "name" and "type" attributes, "policy-rule-deploy-status" and "policy-rule-exec-status" describe the deploy status and execute status of the ECA policy respectively. The most important components are "has-policy-events", "has-policy-conditions" and "has-policy-actions" which denote whether the policy rule has an "event", "condition" and "action" clause. Note that each definition of the clause is in the "supa-policy-statement"

"supa-policy-statement" separates the representation of a SUPA Policy from its implementation. Its subclasses enable the developer to define a SUPA Policy as either a completely reusable set of SUPA Policy objects or as an efficient encoding made up of attributes. "event-clause" defines the policy event the system monitors. If the predefined event is evaluated as TRUE, then the condition will be evaluated. "condition-clause" defines the condition to trigger the corresponding action. It should be a three-tuple clause with {PolicyVariable, PolicyOperator, PolicyValue}. E.g., in "bandwidth utilization > 80% ", "bandwidth utilization" is the variable, ">" is the operator and "80%" is the value. "action-clause" defines the consequence action as the "condition-clause" is evaluated as TRUE.

4.2. Policy Applications in DDC services

More specifically, for the networking system an E-C-A based policy model can cover use cases as follows:

```
Network Policy
    -event:
         -- bandwidth usage
         -- port N status
         -- TTL value
    -condition :
         -- bandwidth usage > x
         -- port N is up
         -- TTL < y
    -action:
         -- adjust flow
         -- use backup link
         -- retry
```

For the first one, if the bandwidth usage of a contain link is above threshold, the flow will be adjusted. The event is the link bandwidth usage, the condition is bandwidth usage above the x and the action is to adjust the flow.

The second one is, if the port N is up, the system will use backup link.

The third one is, if the TTL is below y, retry.

The following describes SUPA policy model designed for DDC services use case [SUPA-DDC] to do the traffic steering among DCs. [SUPA-DDC] took a large-scale Internet Data Center (IDC) operator as an example to describe what SUPA needs to do in VPN-based traffic steering. Here the ECA policy model only focus on the policy based traffic steering application.

The traffic steering policy is used in dynamical link configuration in DDC services [SUPA-DDC]. The service management can dynamically adjust the traffic flow in the links based on traffic steering policy. The policy model specifies some high level requirements to the links, like routing strategy. For example in this specific traffic steering policy model, if the link bandwidth usage is above certain threshold, the monitored flow will be forced to routed to the third place to bypass the overloaded node or site.

Module "ietf-supa-policy" defines E-C-A based policy model to describe the policy management in DDC service. In effect, the module can be expanded with more operation services for DDC services. The ECA model here with the DDC traffic steering application is an instantiation and inheritance of the generic policy model with ECA policy rule.

```
module: ietf-supa-policy
   +--rw supa-ddc-policy
      +--rw adjust-flow-path-policy
         +--rw adjust-flow-path* [vpn-name]
            +--rw vpn-name                          string
            +--rw vpn-type?                         enumeration
            +--rw flow-name?                        string
            +--rw traffic-steering-policy-rule
               +--rw policy-rule-deploy-status?   enumeration
               +--rw policy-rule-exec-status?     enumeration
               +--rw policy-event
               |  +--rw bandwidth?   string
               +--rw policy-condition
               |  +--rw bandwidth-utilization?   enumeration
               |  +--rw operator?                enumeration
               |  +--rw value?                   uint32
               +--rw policy-action-adjust-path
                  +--rw constraint-nodes
                  |  +--rw constraint-node* [nodeId]
                  |     +--rw nodeId              string
                  |     +--ro constraint-type?   enumeration
                  |     +--rw sequence?           uint32
                  +--rw constraint-sites
                     +--rw constraint-site* [siteId]
                        +--rw siteId              string
                        +--ro constraint-type?   enumeration
                        +--rw sequence?           uint32
```

4.2.1. Model for Traffic Steering Policy

```
<CODE BEGINS>
module ietf-supa-policy {
  namespace "urn:ietf:params:xml:ns:yang:ietf-supa-policy";
  // replace with IANA namespace when assigned
  prefix policy;

  import ietf-inet-types {
    prefix inet;
  }

  organization "IETF";
  contact
    "Editor: Jun Bi
     junbi@tsinghua.edu.cn";

  description
    "This YANG module defines a component that describing
```

```
        the ddc policy model for monitoring and optimizing
        tenant's DC (data center) services that are deployed
        between multiple data centers.

        Terms and Acronyms
          DDC: Distributed Data Center
          L2VPN: Layer 2 Virtual Private Network
          L3VPN: Layer 3 Virtual Private Network";

     revision 2015-05-06 {
       description
         "Revised revision.";
         reference
           " Network Policy YANG Data Model ";
     }

   container supa-ddc-policy{
     description
       "Distributed Data Center Service Operation Data";

     container adjust-flow-path-policy {
       description
         "To improve the bandwidth utilization (or reduce the cost,
          or other reason) and mitigate traffic congestion,management
          system/ application requires controller to adjust certain
          flows to pass/bypass certain nodes(or links), when, e.g.,
          bandwidth utilization exceed certain threshold. Vpn name,
          vpn type, adjusted flow and specified nodes (that the flow
          should pass) should be told to controller. So that the
          controller can configure the network elements to change the
          VRF table and routing table. If the link bandwidth usage is
          above certain threshold, the monitored flow will be forced
          to routed to the third place to bypass the overloaded node
          or site.";

       list adjust-flow-path {
         key "vpn-name";
         description
           "The list of VPN and flow that need to be adjusted in
            specific paths. So as to optimize traffic in the links
            that are between data centers.";
         leaf vpn-name {
           type string;
           mandatory true;
           description
             "Indicates the name of VPN that the adjusted flow
              belongs to. A VPN instance is identified by vpn-name.
              It should be one of the created VPN instance names";
```

```
        }
        leaf vpn-type {
          type enumeration {
            enum L2VPN {
              description "L2VPN";
            }
            enum L3VPN {
              description "L3VPN";
            }
          }
          description "Indicates the type of VPN instance that the
          adjusted flow belongs to. L2VPN or L3VPN";
        }
        leaf flow-name {
           type string;
           description "The name of the adjusted flow. So as to tell
           the Controller which flow should be adjusted";
        }

      container traffic-steering-policy-rule{

        leaf policy-rule-deploy-status {
          description "It defines the current deployment status
          of this policy rule. Both operational and test mode values
          are included in its definition.";
             type enumeration {
               enum 0{
                description "undefined";
               }
               enum 1{
                description "deployed and enabled";
               }
             enum 2{
                 description "deployed and in test";
               }
             enum 3{
                 description "deployed but not enabled";
               }
             enum 4{
                 description "ready to be deployed";
               }
             enum 5{
                 description "not deployed";
               }
            }
           }
```

```
        leaf policy-rule-exec-status {
          description "It defines the current execution status
          of this policy rule. Both operational and test mode values
          are included in its definition";
             type enumeration {
               enum 0{
                description "undefined";
               }
               enum 1{
                description "executed and SUCEEDED (operational
                mode)";
               }
               enum 2{
                description "executed and FAILED (operational
                mode)";
               }
               enum 3{
                description "currently executing (operational
                mode)";
               }
               enum 4{
                description "executed and SUCEEDED (test mode)";
               }
               enum 5{
                description "executed and FAILED (test mode)";
               }
               enum 6{
                description "currently executing (test mode)";
               }
             }
           }

        container policy-event{
          description "The Policy defines an ECA type policy
          model for the service management of traffic optimization
          between DCs. If the predefined event is monitored, then
          the controller will evaluate if the condition is matched.
          If the condition is matched, the corresponding action will
          be performed. Here in this case, bandwidth is the 'event'
          to monitor in the traffic steering policy case";

          leaf bandwidth {
            type string;
          }
        }

        container policy-condition {
```

```
            description "There are several types of 'condition'
            match, here it is above/below";
            leaf bandwidth-utilization {
              type enumeration {
                   enum utilization {
                   description "In this case, the bandwidth
                   utilization is the 'event'. The controller will
                   monitor the bandwidth utilization if it is above
                   the threshold which is the link utilization ratio,
                   0-100%";
                   }
                   enum BW {
                   description " In this case, the bandwidth usage
                   is the 'event'. The controller will monitor the
                   bandwidth usage if it is above the threshold which
                   is bandwidth value, 2G,10G e.g.";
                }
              }
            }

            leaf operator{
              type enumeration {
                   enum above {
                   description "If the value of the monitored event
                   is above the threshold, do the action. E.g., in
                   this case, if the bandwidth utilization is above
                   70%, 'above' is the 'match-type' here.";
                   }
                   enum below {
                   description "If the value of the monitored event
                   is below the threshold, do the action. E.g., in this
                   case, if the bandwidth usage is below 2Gb, 'below'
                   is the 'match-type' here.";
                }
              }
            }

            leaf value {
              description "The target value of the monitored
              'event' or the threshold to trigger the action. E.g.,
              in this case, if the bandwidth utilization is above 70%,
              '70%' is the 'value' here. If the bandwidth usage is
              above 2Gb, here '2Gb' is the 'value' here";
              type uint32;
            }
          }

          container policy-action-adjust-path {
```

```
            description "This is the corresponding 'action' when
            the condition has been triggered. E.g., if the link
            occupied ratio is above 70%, adjust the path of the flow.
            Here the 'adjust-path' is the action of the ECA policy. If
            the 'event' is monitored, and the 'condition' is triggered,
            the corresponding 'action', rerouted the flow path, will
            be performed.";

            container constraint-nodes {
                description "The generated 'action' of the ECA
                policy. It is a description of the constraints on the
                nodes those need to be adjusted. Here the 'action' only
                describe the constraint not the real operation. And
                then, the controller will compute a new route based on
                the constraint. More specifically, the constraint-nodes
                are a set of nodes that specify the constraints by the
                up-level. Each node in the list will be passed/bypassed
                when computing the new route.";
              list constraint-node {

                max-elements unbounded;
                min-elements 0;
                description "There can be a set of nodes to be
                constrained. Each node in the list will be passed/
                bypassed when computing the new route.";
                key nodeId;
                description " List of nodes that the adjusted
                flow needs to pass or bypass so as to adjust the flow
                path between data centers. E.g. if the event and
                condition has been triggered that the vpn flow will be
                adjusted. The constraint-nodes specify how to adjust
                the flow and which nodes will be adjusted. Each node
                in the list will be told to controller to pass/bypass
                when computing the new route.";
                leaf nodeId {
                  description "The node needs to be constrained
                  that describe which nodes will be adjusted in the
                  path. The node ID is the identifier of the node that
                  the controller will take into account when computing
                  the new route.";
                  config true;
                  type string {
                  length "0..64";
                  pattern "([^?]*)";
                  }
                }
                leaf constraint-type {
                  description "The node can be bypassed or
                  passed.";
                  config false;
```

```
                type enumeration {
                  enum bypass {
                    value 0;
                    description "The node will be bypassed.
                    E.g., if the corresponding action of the ECA
                    policy is to bypass the busy node 1, 2 and 3,
                    then the node-list will be nodeId: 1, 2 and 3,
                    while the'constraint-type' of each node is
                    'bypass' which denote all these nodes should be
                    bypassed when computing the new route. Then the
                    controller will compute the new route based on
                    the constraint to compute a new path without
                    node 1, 2
                    and 3.";
                  }
                  enum pass {
                    value 1;
                    description "The node will be passed. E.g., if
                    the corresponding action of the ECA policy is to
                    pass the light loaded nodes 1, 2 and 3, then the
                    node-list will be nodeId: 1, 2 and 3, while the
                    'constraint-type' of each node is 'pass' which
                    denote all these nodes should be passed when
                    computing the new route. Then the controller
                    will compute the new route based on the
                    constraint to compute a new path without node
                    1, 2 and 3.";
                  }
                }
              }
              leaf sequence {
                description "Constraint node sequence. The
                corresponding action can also specify the sequence
                of the node list that should be passed/bypassed.
                E.g., if the 'action' is to adjust the flow to a
                new path with node 1, 2 and 3, while the pass the
                node 1 first, then 3, finally 2. Here the sequence
                of the node-list will be 1-3-2. Then the controller
                will compute the new path based on the sequence.";
                config true;
                default 1;
                type uint32 {
                  range "0..128";
                }
              }
            }
          }
          container constraint-sites {
            list constraint-site {

              max-elements unbounded;
              min-elements 0;
              description ".";
              key siteId;
```

```
              description " List of sites that the adjusted
              flow needs to pass or bypass So as to adjust the flow
              path between data centers. The site can be a group of
              nodes at different granularities for example data
              centers.";

              description " List of sites that the adjusted
              flow needs to pass or bypass So as to adjust the flow
              path between data centers.";

              leaf siteId {
                description "The site needs to be adjusted in
                the flow path. The site";
                config true;
                type string {
                length "0..64";
                pattern "([^?]*)";
                }
              }
              leaf constraint-type {

                description "The constraint type of the site
                to specify whether the node site will be passed or
                bypassed";
                config false;

                type enumeration {
                  enum bypass {
                    value 0;
                   description "The site will be bypassed";
                  }
                  enum pass {
                    value 1;
                    description "The site will be passed";
                  }
                }
              }

              leaf sequence {
                description "constraint site sequence";
                config true;
                default 1;
                type uint32 {
                range "0..128";
                }
              }
            }
          }
        }
```

```
        }
       }
      }
    }
  }
  <CODE ENDS>
```

4.2.2. Policy Based Traffic Steering Operation

The associated action generated by the ECA model is sent to the controller such as the list of constraint nodes which denotes the constraint of the adjusted flow path. E.g. the node C will be bypassed.

Based on the steering information, the Network Manager/Controller generates a path which meets the requirements: e.g., the computed path is (A, D, E, B). Network Manager/Controller also has to configure each device on the new path, not only the devices specified by the configuration such as node D, but also the devices in the underlying network which must be reconfigured, such as node E. The topology information is also necessary when Network Manager/Controller decides which device ought to be configured.

5. Security Considerations

TBD

6. IANA Considerations

This document has no actions for IANA.

7. Contributor List

Andy Bierman

YumaWorks, Inc.

Email: andy@yumaworks.com

8. Acknowledgments

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in alphabetical order: Jing Huang, Junru Lin, Felix Lu, Juergen Schoenwaelder, Yiyong Zha, and Min Zha.

9. References

9.1. Normative References

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC3272] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and X. Xiao, "Overview and Principles of Internet Traffic Engineering", RFC 3272, May 2002.

9.2. Informative References

[SUPA-framework] C. Zhou, L. M. Contreras, Q. Sun, and P. Yegani, " The Framework of Shared Unified Policy Automation (SUPA) ", IETF Internet draft, draft-zhou-supa-framework, January 2015.

[SUPA-problem-statement] G. Karagiannis, Q. Sun, Luis M. Contreras, P. Yegani, and JF Tremblay, "Problem Statement for Shared Unified Policy Automation (SUPA)", IETF Internet draft, draft-karagiannis-supa-problem-statement, January 2015.

[SUPA-DDC] Y. Cheng,and JF. Tremblay, "Use Cases for Distributed Data Center Applications in SUPA", IETF Internet draft, draft-cheng-supa-ddc-use-cases, January 2015.

[RESTCONF] Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando, "RESTCONF Protocol", draft-ietf-netconf-restconf (work in progress), July 2014.

[POLICY MODEL] Z. Wang, L. Dunbar, Q. Wu, "Network Policy YANG Data Model" draft-wang-netmod-yang-policy-dm, January 2015.

Authors' Addresses

Jun Bi
Tsinghua University
Network Research Center, Tsinghua University
Beijing 100084, China
Email: junbi@tsinghua.edu.cn

Raghavendra Tadepalli
Wipro Limited
Email: raghav.rao@wipro.com

Michiaki Hayashi
KDDI R&D Labs. Inc.
2-1-15 Ohara, Fujimino, Saitama, Japan. 356-8502
Email: mc-hayashi@kddilabs.jp

Network Working Group
Internet-Draft
Intended status: Informational
Expires: December 30, 2015

Y. Cheng
China Unicom
D. Zhang
Aliababa Group
JF. Tremblay
Viagenie
S. Zhu
Alibaba Group
J. Bi
Tsinghua University
L. M. Contreras
Telefonica I+D
June 30, 2015

Use Case for Distributed Data Center in SUPA
draft-cheng-supa-ddc-use-cases-08

Abstract

Large scale Distributed Data Centers (DDCs) can provide various services and usually consist of numbers of internal and external links where various VPNs are built upon. The Service provisioning and network connectivity configurations could be complex and dynamic, and manual configuration is onerous and error-prone. This draft analyzes the use cases in DDCs, in which some VPN scenarios are covered, and the applicability of Simplified Use of Policy Abstractions (SUPA) data models which can be used for better and automated resource usage and easy service/network deployment/ configuration.

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 December 6, 2015.

Table of Contents

1. Introduction

The SUPA (Simplified Use of Policy Abstractions) work aims at providing data models, including network service data models, policy data models, to easily, accurately, and efficiently select and use the available communication network capabilities. An example of the data model can be found in [I-D.zaalouk-supa-vpn-service-management-model]. Service Manager (SM) is used by an a communications service provider and/or operator to deploy and manage services on top of network facilities. An example of SM is a set of applications used by an Operational Support System (OSS) entity to perform network configuration. Several SUPA

use cases have been introduced in the problem statement document. This document reviews various scenarios for Distributed Data Center (DDC) use case.

A large-scale Distributed Data Center (DDC) operator may have to maintain multiple data centers and the inter-connecting networks in order to provide server hosting, leased line, and value-added services to enterprises and ISPs, and have multiple data centers. In this DDC network, traffic at each site is routed via configuring policy routes and the operator may be able to adjust routes prioritization to choose an outgoing link. This type of static provisioning comes with high costs and poor operability，and as a result the bandwidth resources intra/inter-data centers are not efficiently utilized.

In quite a few scenarios, the links between DCs are VPNs (MPLS, SSL, IPsec, and etc.). SUPA will be mainly used for those VPN configurations. Although there may be some cases where physical links are used, but those are out of the scope of this draft.

DC and network may belong to different operators. If a DC operator needs to configure network connectivity for DCs, it may need to cooperate with network operators providing such connectivity. Network operators can define and provide data models to enable this.

This document illustrates several distributed datacenter (DDC) applications and explains how an operator could use SUPA to provide these applications.

2. Requirements Language

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 [RFC2119].

3. Terminology

The terminology used in the SUPA problem statement draft [I-D.zhou-supa-framework] and [I-D.karagiannis-supa-problem-statement] apply also to this draft.

DC Data Center

DDC Distributed Data Center

ECA Event Condition Action

NM/NC Network Manager / Controller

OSS	Operational Support System
SM	Service Manager
SUPA	Simplified Use of Policy Abstractions
TTM	Time to Market
VAS	Value Added Service
vDC	virtual Data Center
VPC	Virtual Private Cloud (PC)

4. Challenges Faced by Data Center ISPs

There are many challenges in traditional DDCs:

1. The infrastructure and network links rent from other operators, are normally dependent on manual planning and design, which leads to inflexibility and low efficiency of resource usages. SUPA can benefit the DDC operators to manage the network resources rent from others in a more flexible way.

2. Service expansion is limited in a single physical DC. Each DC resource is isolated, so service and resource can only be deployed in one single DC.

3. VAS (Value Added Service) is provisioned via static configuration, which brings complex training, long service TTM time and poor flexibility. This could not meet the requirements of complex use cases, e.g., lot of VAS devices, significant differences between various services.

5. SUPA Benefits

There are requirements from DDC operators to optimize and automate the service deployment. Also, a tenant may expect the traffic steering capabilities in order to use the rent links more efficiently. SUPA can show its advantages in above cases since it is able to:

- Support an open network architecture: standardizaed data models enable an open architecture and make it possible for unified service / network planning, which can interconnect with third party cloud platform, supporting fast service innovation.

- Support overall DC resource integration: SUPA data models can be used for network resource virtualization; inter-DC resource, virtual DC (vDC) resource, etc, can be integrated and controlled by a centralized functional entity.

- Support automatic E2E service delivery: Network (including virtual network), computing, inter-DC management of storage resource, automatic service delivery, automatic VPN connection configurations between DCs, which improves operation efficiency.

- Improve DDC network usage by means of Intelligent scheduling of DDC traffic, improving link usage.

- Support VAS service on-demand provisioning automatically: Create or delete VAS nodes on-demand, based on various service requirements; network forwarding policy based on the VAS routing, to achieve automatic draining and automatic configuration of VAS device policy.

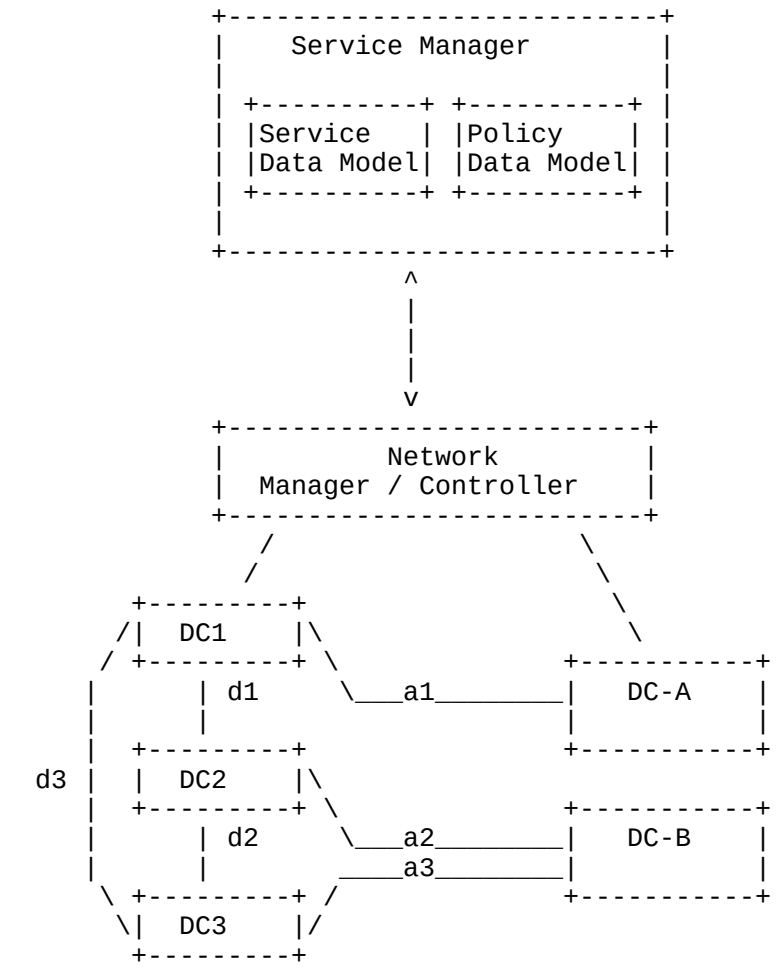
Please refer to [I-D.zhou-supa-framework] and other SUPA related documents for more details of SUPA features.

The following sections will illustrate three typical cases in distributed data center which could benefit from SUPA architecture.

6. Scenarios

In the following uses, Service Manager (SM) is used for service and policy definition; and Network Manager (Controller) is used for network topology maintenance and mapping data models to detail network configurations, as defined in [I-D.zhou-supa-framework].

6.1. Scenario:Inter DC Connectivity



Inter DC Connectivity

There can be a number of links between data centers, and the configuration of such links could complex. As shown in Figure 1, service data models and policy data models can be defined to automate the configuration procedures. The service data model for connectivity is used for specifying attributes of (virtual) links, e.g. the end points of links, bandwidth, QoS and availability parameters, etc. The policy model can specify some high level requirements to the links, like routing strategy (via and not via) and price/cost strategy. The policy data model can also define the policy rules that drive the security requirements.

An application example is, the links interconnecting two DCs together should guarantee a minimum bandwidth, certain QoS parameters, and provide availability guarantees.

Anothe service policy example in Figure 1, for traffic from DC2 to DC-B, if the load on a link exceeds a threshold (e.g., 90%), some (new) traffic can be redirect to another link.

Requirements and configurations derived from above application scenarios can be described by service data model and policy data model.

```
module: ietf-supa-ddc
   +--rw ddc-services
      +--rw ......                  other possible attributes
      +--rw ddc-service* [name]
      |  +--rw name                 string
      |  +--rw connection-type?     enumeration
      |  +--rw connection-name      string
      |  +--rw bandwidth            uint32
      |  +--rw latency              uint32
      +--rw ......                  other possible attributes
```

Service Data Model for Inter DC Connectivity

The above service data model can be used to describe links attributes for a VPN, including bandwidth , latency, etc.

```
module: ietf-supa-policy
   +--rw supa-policy
      +--rw ......                          other possible attributes
      +--rw supa-policy-atomic
      |  +--rw supa-ECA-policy-rule
      |     +--rw policy-rule-name?         string
      |     +--rw has-policy-events?        boolean
      |     +--rw has-policy-conditions?    boolean
      |     +--rw has-policy-actions?       boolean
      +--rw ......                          other possible attributes
```

ECA Policy Data Model

The above policy data model can be used to describe the requirement that when the load on a link exceed a threashold. In this case,

"event" is the bandwidth of a link, "condition" is "load >= 80%", "action" is "redirect some traffic to another link".

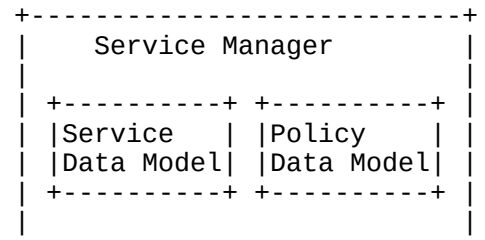
```
module: ietf-supa-policy
   +--rw policy-set
      +--rw ......                          other possible attributes
      +--rw policy-rule
         +--rw rule-name?                   string
         +--rw rule-type?                   enumeration
         +--rw policy-rule-priority?        uint8
         +--rw intent-policy-rule
         |  +--rw desired-state
         |  |  +--rw constraint?            string
         |  |  +--rw constraint-priority?   uint8
         |  +--rw behavior-constraint
         |     +--rw constraint?            string
         |     +--rw constraint-priority?   uint8
         +--rw ......                       other possible attributes
```

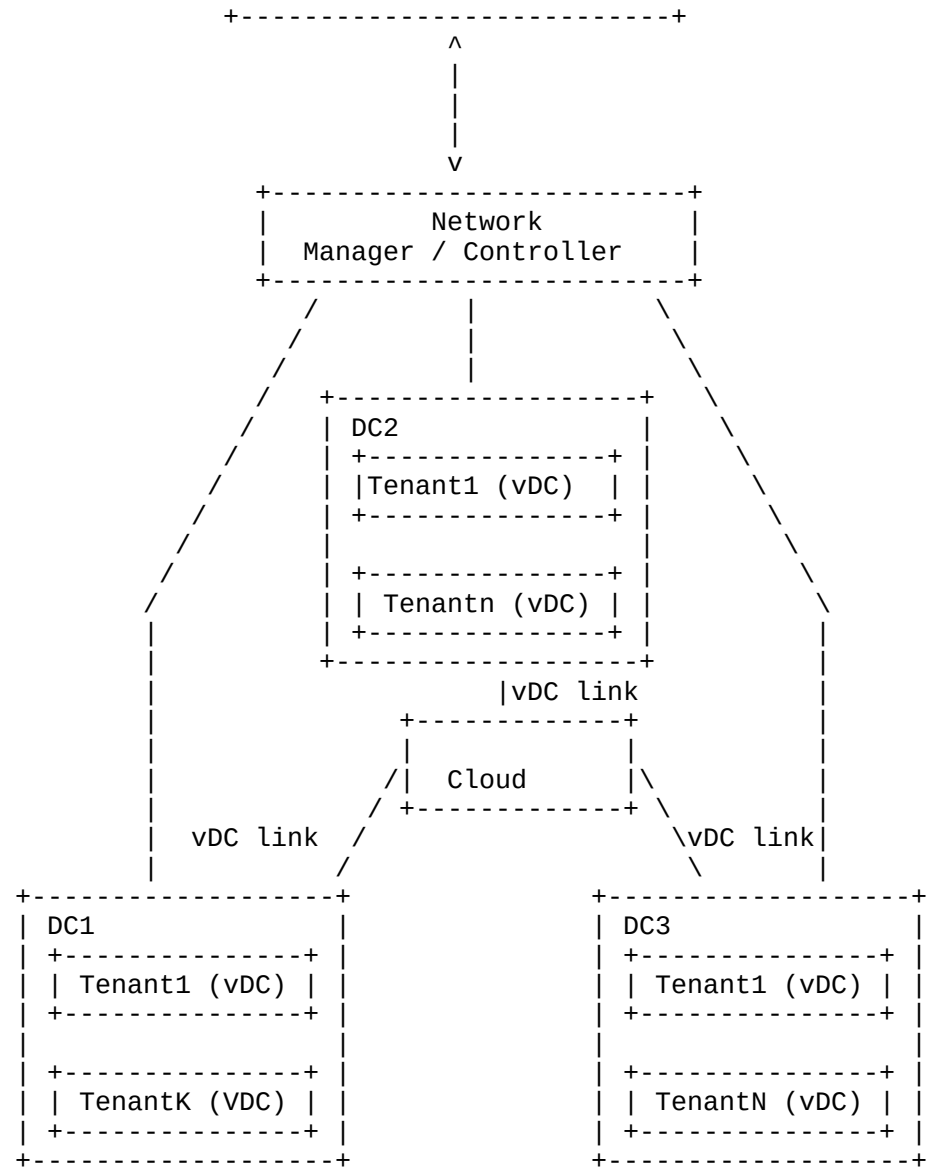
Policy Data Model for high level requirements

The policy model shown above can be used to express some sophisticated requirements, e.g. the number of hop of any link should be less than 5, or any links should not share any network nodes in between and should be completely independent to each other so as to achieve high availablity in case of network node failure.

Inter DC connections can be classified into two types: connections within a single administrative domain and connections across multiple administrative domains. Links d1, d2 and d3 are within an administrative domain; and links a1, a2 and 3 are across domains. The difference between them is that connections across multiple administrative domain require extra negotiation and authentication/ authorization, which can be achieved via communications between SMs. Data models for this purpose should also be defined.

6.2. Scenario:vDC Connectivity





vDC Connectivity

A DC tenant may have resources, e.g. network, computing, storage, etc, in multiple physical DCs. DC operators will provide internal network connectivity for these distributed resources, and make it

look like one seamless entity, which can be called as virtual DC (vDC).

The internal links for vDC can be implemented via tunneling overlay technologies, e.g. VPN or VxLAN, etc. The tunnels need to be dynamically established, managed and released.

As show in Figure 5, service data model and policy data model can be defined to automate the links configuration for vDCs. A service data model should specify the attributes of the tunnels, e.g., bandwidth, QoS and availability parameters. Policy systems can dynamically scale the DC resources assigned to a tenant, and the policy rules that drive the prioritization of resource assignments. The networking resources assigned to a tenant should scale proportionally to the compute resources assigned to a tenant. The traffic should be prioritized to resources owned by tenants that offer interactive services according to the time zone the DC is located in. Because a DC serving enterprise may require higher priority in working hour, and a DC providing entertainment service may need higher network priority in non-working hours.

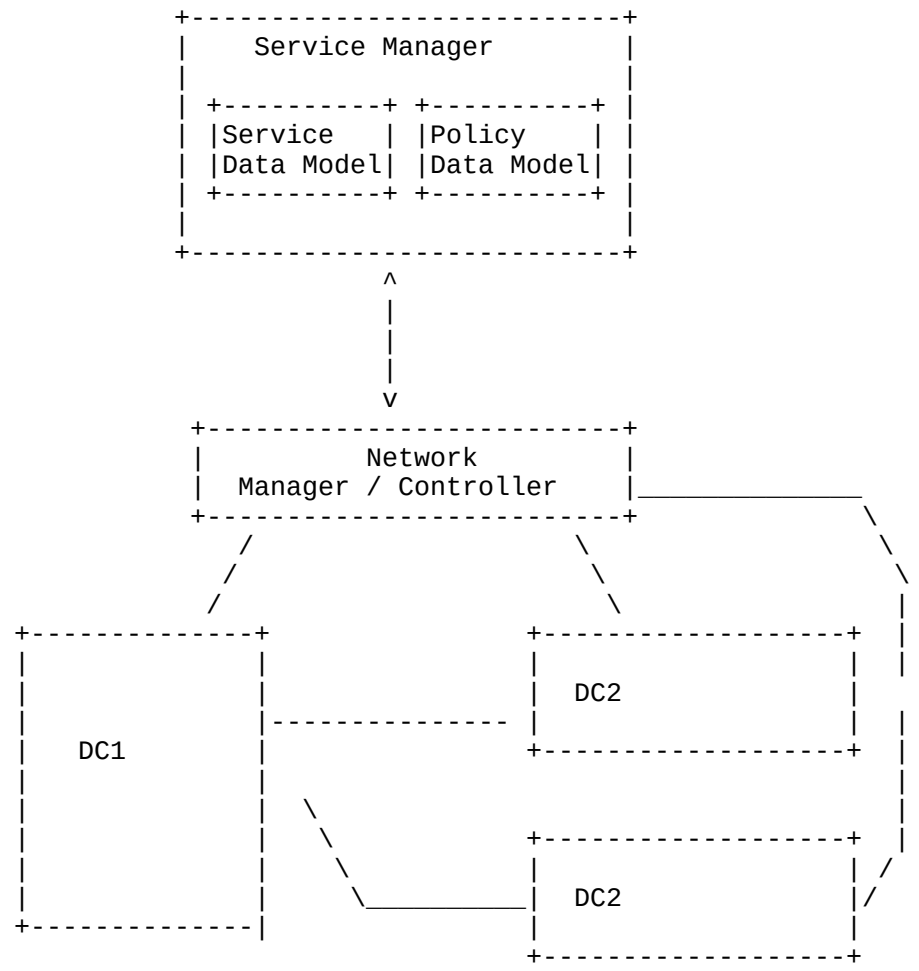
```
module: ietf-supa-policy
   +--rw supa-policy
      +--rw ......                          other possible attributes
      +--rw supa-policy-validity-period
      |  +--rw start?                       yang:date-and-time
      |  +--rw end?                         yang:date-and-time
      |  +--rw duration?                    uint32
      |  +--rw periodicity?                 enumeration
      +--rw supa-policy-atomic
      |  +--rw supa-ECA-policy-rule
      |     +--rw policy-rule-name?         string
      |     +--rw has-policy-events?        boolean
      |     +--rw has-policy-conditions?    boolean
      |     +--rw has-policy-actions?       boolean
      +--rw ......                          other possible attributes
```

Policy Data Model for vDC Connectivity

In the above policy data model, events and conditions may not be necessary; the actions will be based on the time; and two actions will be required: set the VPN priority to low or high.

6.3. Scenario:Dynamic Link Configuration for DC

Static and over provisioning for DC links is not always a good solution. Sometimes dynamic configuration is necessary.



Dynamic Link Configuration for DC

One case is virtual machine migration and large amount of data transfer between DCs. But this kind of activity does not happen frequently. A dedicated link with constant bandwidth for this purpose is too expensive. The network operator should allow the DC operator to create a link on demand when necessary. This link may

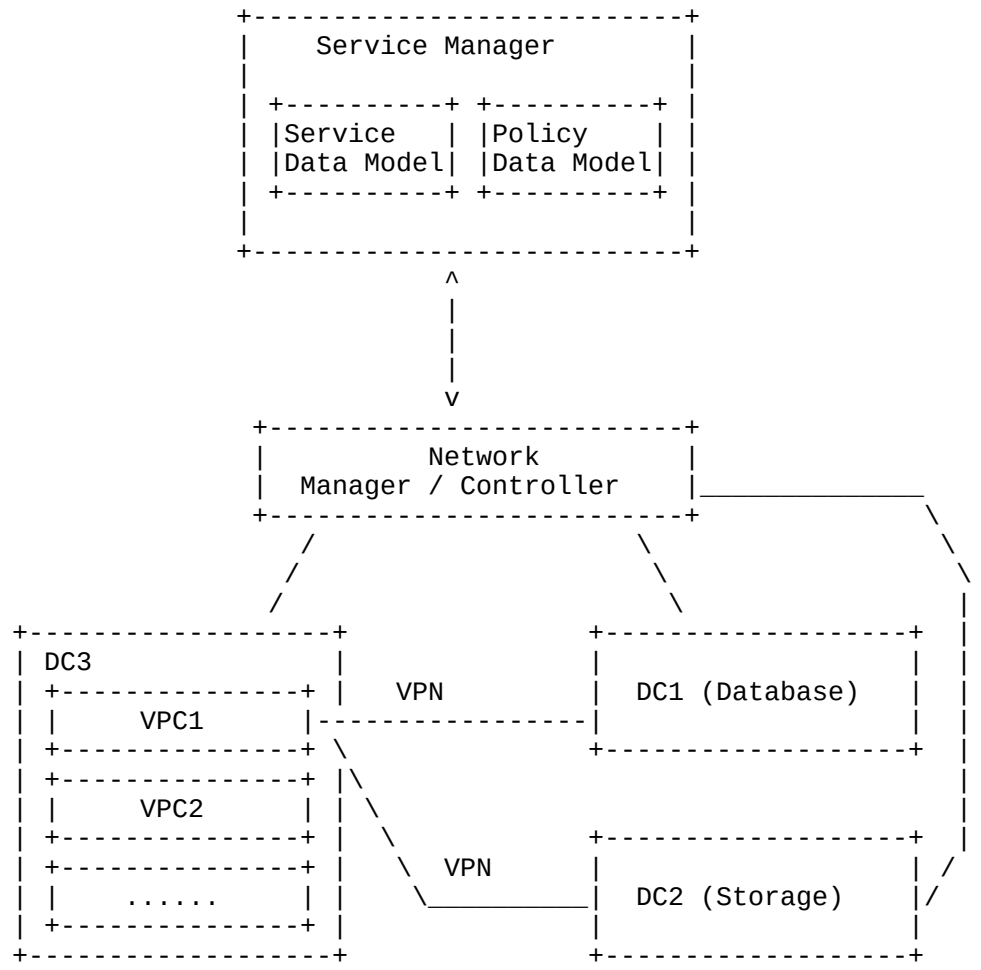
have large bandwidth but last for a limited time period. An alternative is to create short-term dedicated links for backups and migrations.

As shown in Figure 7, data models can help to automate these kind of configurations. In the data models, the attributes of links (bandwidth, QoS and availability parameters) should be specified. The policy concerning strict and soft bounds on the lifetime of such links, and the policy concerning the scheduling of dedicated links (e.g., based on the current load) and the services using the dedicated links can also be specified.

When the traffic volume between DCs exceeds a certain threshold, the policy-driven service manager requests that traffic schedules may be adjusted within bounds in order to balance load on the links (e.g., delay backups and migrations until the network has the necessary capacity).

In this case, the ECA policy model will apply, but the action is different -- change the bandwidth of link(s) with time period constraints, as shown in Figure6.

6.4. Scenario:DC Connectivity for Virtual Private Clouds (VPC)



VPC to DC Connectivity

In practice, a public cloud operator can virtualize its cloud resources into different isolated virtualized private clouds and provided them for 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 the control portal, a tenant can establish and manage its network easily, for instance, deploying or removing virtualized network devices (e.g., virtualized routers and virtualized switches), adjusting the topology of VPC networks,specifying packet forwarding policies, and deploying or un-deploying virtual services (e.g., load balancers, firewalls, databases, DNS, etc.). The network functionalities that the tenant can accessed are virtualized and actually 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 newly deploys two VMs in its 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 many cases, a tenant may need to specify how the VPCs is connected to its enterprise cloud networks. For instance, a tenant may want 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, 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 the database services only when the tenant has deployed database into its VPC through the control portal.

As shown in figure 4, service data models and policy data models can be defined to automate the configurations of links between VPC and DC where service is located. The data models should specify the policy controlling authentication and authorization concerning access to data residing in internal services.

7. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting of detailed configuration states. More investigation remains to fully define the security requirements, such as authorization and authentication levels.

8. IANA Considerations

Not applicable.

9. Acknowledgements

The authors of this draft would like to thank the following persons for review, discussion, and valuable comments: Cathy Zhou, Georgios Karagiannis, Scott O. Bradner, James Huang, Bob Natale.

10. Contributors

The following persons contribute use case and text to this draft, and are listed below:

Scott O. Bradner
sob@sobco.com

Dacheng Zhang
dacheng.zdc@alibaba-inc.com

11. References

11.1. Normative References

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

11.2. informative References

[I-D.karagiannis-supa-problem-statement]
Karagiannis, G., Qiong, Q., Contreras, L., Yegani, P., and J. Bi, "Problem Statement for Simplified Use of Policy Abstractions (SUPA)", draft-karagiannis-supa-problem-statement-06 (work in progress), March 2015.

[I-D.zaalouk-supa-vpn-service-management-model]
Zhang, D., Zaalouk, A., Pentikousis, K., and Y. Cheng, "VPN Service Management YANG Data Model", draft-zaalouk-supa-vpn-service-management-model-03 (work in progress), April 2015.

[I-D.zhou-supa-framework]
Zhou, C., Contreras, L., Qiong, Q., and P. Yegani, "The Framework of Simplified Use of Policy Abstractions (SUPA)", draft-zhou-supa-framework-01 (work in progress), February 2015.

Authors' Addresses

Ying Cheng
China Unicom
P.R. China

Email: chengying10@chinaunicom.cn

Dacheng Zhang
Aliababa Group

Email: dacheng.zdc@alibaba-inc.com

JF Tremblay
Viagenie

Email: jean-francois.tremblay@viagenie.ca

Shunmin zhu
Alibaba Group

Email: jianghe.zsm@taobao.com

Jun Bi
Tsinghua University
Bei Jing
China

Email: junbi@cernet.edu.cn

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com

Network Working Group
Internet Draft
Intended status: Informational
Expires: December 5, 2015

G. Karagiannis
J. Strassner
Huawei Technologies
Q. Sun
China Telecom
Luis M. Contreras
Telefonica
P. Yegani
Juniper Networks
J.Bi
Tsinghua University

June 5, 2015

Problem Statement for Simplified Use of Policy Abstractions (SUPA) draft-karagiannis-supa-problem-statement-07

Abstract

The increase in complexity of modern networks makes it challenging to deploy new services and to keep networks up to date whilst maintaining stability and availability for critical business services. This is a major challenge that network operators (service providers, SME, etc) face today. The operators aim of streamlining both operations and the deployment of new services, is being met by increasingly relying on (1) software abstractions to simplify the design and configuration of monitoring and control operations and (2) the use of programmatic control over the configuration and operation of such networks.
In this context, providing network operators with a generic policy-based management model that can be used to express policies on top of arbitrary configuration data models is essential.

In particular, SUPA addresses the needs of operators and application developers to use a generic policy-based management model for defining and representing multiple types of policy rules.

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."

Table of Contents

1. Introduction

Network operators are faced with networks of increasing size and complexity while trying to improve their quality and availability, as more and more business services depend on them. Software abstractions to simplify the design and configuration of monitoring and control operations and the use of programmatic control, often called software-defined, are considered by many network operators as an essential tool toward the management of that complexity.

Providing means to network operators to (1) express policies on top of arbitrary configuration data models and (2) represent multiple types of policy rules, enable significant improvements in configuration agility, error detection and uptime for operators. This document describes the problems that need to be addressed in order to equip service providers with the means, such as a generic policy-based management model used to represent multiple types of policy rules to quickly and dynamically manage their offering of network services.

1.1 Motivation

The rapid growth in the variety and importance of traffic flowing over increasingly complex enterprise and service provider network makes the task of network operations and management applications and of deploying new services much more difficult.

This is a significant challenge that network operators (service providers, SME, etc) face today.

Several mechanisms can be used to deal with this challenge. The main ones are: (1) the use of software abstractions to simplify the design and configuration of monitoring and control operations and (2) the use of programmatic control over the configuration and operation of such networks. By combining these mechanisms can (1) provide additional and significant benefits in design and deployment agility and (2) be used to define a generic policy-based management model.

In particular, the power of policy management is its applicability to many different types of systems. Many different types of actors can be identified that can use a policy management system, including applications, end-users, developers, network administrators and operators. Each of these actors, typically, has different skills and uses different concepts and terminologies. For example, an operator may want to express that only Platinum and Gold users can use streaming and interactive multimedia applications. As a second example, an operator may want to define a more concrete policy rule that looks at the number of dropped packets. If, for example, this number exceeds a certain threshold value, then the applied queuing, dropping and scheduling algorithms could be changed in order to reduce the number of dropped packets.

Both examples can be referred to as "policy rules", but they take very different forms, since they are at very different levels of abstraction and likely authored by different actors. The first example described a very abstract policy rule, and did not contain any technology-specific terms, while the second example included a more concrete policy rule and likely used technical terms of a general (e.g., IP address range, port numbers) as well as vendor-specific nature (e.g., specific algorithms implemented in a particular device). Furthermore, these two policy rules could affect each other. For example, Gold and Platinum users might need different device configurations to give the proper QoS markings to their streaming multimedia traffic. This is very difficult to do if a common policy framework does not exist.

It needs to be mentioned that there are ongoing policy modeling efforts in IETF. However, all these policy modeling models can be characterized as being technology specific. This means that the IETF needs to reinvent the wheel in different colors (i.e., policy models that apply for a specific technology) several times.

SUPA will address these challenges by:

(1) developing an information model fragment for defining standardized policy rules at different levels of abstraction,
(2) specifying how to map this information fragment into corresponding YANG [RFC6020] and [RFC6991], data models to define interoperable implementations that can exchange and modify generic policies using protocols such as NETCONF/RESTCONF on the interface north of the controller (or other similar management entity) and south of the service manager.

Specifically, three information model fragments are envisioned:

(a) a generic policy information model (GPIM) that defines concepts needed by policy management independent of the form and content of the policy

(b) a more specific information model that refines the generic information model to specify how to build policy rules of the event-condition-action (ECA) paradigm

(c) a more specific information model that refines the generic information model to specify how to build policy rules that declaratively specify what goals to achieve (but not how to achieve those goals); this is often called "intent-based" policy

2. Terminology

Some of definitions are based on [RaBe11] and/or [Stras02].

Network Service: the composition of network functions as defined by its functional and behavioral specification. A network service is characterized by performance, dependability, and security specifications. Furthermore, a network service is delivered by network service endpoints, which may be aggregations of multiple lower-layer technology specific endpoints.

Network Element: a physical or virtual entity that implements one or more network function(s). NEs can interact with local or remote network controllers in order to exchange information, such as configuration information and status.

Service specific abstraction: an abstract view of the service topology, associated with a specific network service type, e.g., inter-datacenter communication services

Policy: a definite goal, course, or method of action to guide and determine present and future decisions. Policies are implemented or executed within a particular context.

SUPA policy: a means to monitor and control the changing and/or maintaining of the state of one or more managed entities.

Policy-based management: the usage of policy rules to manage one or more entities.

Information Model: a representation of managed objects and their relationships that is independent of data repository, language, and protocol.

Data Model: a representation of managed objects and their relationships that is dependent on data repository, language, and/or protocol (typically all three).

Policy Rule: A container that uses metadata to define how the content is interpreted, and hence, how the behavior that it governs is defined separates the content of the policy from its representation provides a convenient control point for OAMP operations.

Policy condition: a representation of the necessary state and/or prerequisites that define whether a policy rule's actions should be performed.

Policy action: defines what is to be done to enforce a policy rule when its conditions are met.

Event Condition Action policy: reactive behavior of a system that correlates a set of events, a set of conditions, and a set of actions. Conditions are evaluated on the occurrence of an event and which determine whether the policy is applicable or not for that particular situation. Furthermore, the actions are only executed only if the conditions are met.

Goal (Intent) policy rule (also called a declarative policy rule, or an intent-based policy rule): a declarative statement that defines what the policy should do, but not how to implement the policy.

Model Mapping: a translation from one type of model to another type of model. Model mapping changes the representation and/or level of abstraction used to another representation and/or level of abstraction. The most common form of model mapping is from an information model to a data model; another important form is from a vendor-neutral data model to a vendor-specific data model.

3. Use Case: Distributed Data Centers (DDCs)

Large scale Distributed Data Centers (DDCs) can provide various services and usually consist of many internal and external links where various VPNs are built upon. The Service provisioning and network connectivity configurations could be complex and dynamic, for which manual configuration is not onerous and error-prone.
The SUPA generic policy management models can be used to support the dynamic and automated resource usage and simplify and automate the

service/network deployment/configuration of various VPN scenarios in the DDC environment. A more detailed description of this use case is provided in [ID.draft-cheng-supa-ddc-use-cases].

4. Requirements

In order to satisfy the challenges mentioned in Section 1.1 and the goal of the DDC use case briefly described in section 3, the following requirements need to be addressed:

- o) Specify a generic and non-technology specific policy information model.
- o) Specify a more specific information model that defines how to build policy rules of the event-condition-action (ECA) paradigm.
- o) Specify a more specific information model that defines how to build policy rules that declaratively specify what goals (what intents) to achieve using the Goal (Intent) policy paradigm.
- o) Specify how to map the above mentioned information models into corresponding YANG standardized data models to define interoperable implementations that can exchange and modify generic policies using protocols such as NETCONF/RESTCONF on the interface north of the controller (or other similar management entity) and south of the service manager.

5. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting detailed configuration states of network elements. This places additional security requirements on SUPA (e.g., authorization, and authentication of network services) that needs further investigation. Moreover, policy interpretation can lead to corner cases and side effects that should be carefully examined, e.g., in case policy rules are conflicting with each other.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgements

The authors of this draft would like to thank the following persons for the provided valuable feedback and contributions: Diego Lopez, Spencer Dawkins, Jun Bi, Xing Li, Chongfeng Xie, Benoit Claise, Ian Farrer, Marc Blancet, Zhen Cao, Hosnieh Rafiee, Mehmet Ersue, Simon Perreault, Fernando Gont, Jose Saldana, Tom Taylor, Kostas Pentikousis, Juergen Schoenwaelder, John Strassner, Eric Voit, Scott O. Bradner, Marco Liebsch, Scott Cadzow, Marie-Jose Montpetit.

Tina Tsou, Will Liu and Jean-Francois Tremblay contributed to an early version of this draft.

8. References

8.1. Normative References

8.2. Informative References

[ID.draft-cheng-supa-ddc-use-cases] Y. Cheng, JF. Tremblay, J. Bi, L. M. Contreras, "Use Case for Distributed Data Center in SUPA", IETF Internet draft (Work in progress), draft-cheng-supa-ddc-use-cases-07, May 8, 2015

[RFC6020] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6991] J. Schoenwaelder, "Common YANG Data Types", RFC 6991, July 2013.

[RaBe11] Raphael Romeikat, Bernhard Bauer, "Formal Specification of DomainSpecific ECA Policy Models", in Proc. 2011 Fifth IEEE International Conference on Theoretical Aspects of Software Engineering, 2011

[Stras02] John Strassner, "DEN-ng: Achieving Business-Driven Network Management" in Proc. IEEE Network Operations and Management Symposium (NOMS), 2002.

Authors' Addresses

Georgios Karagiannis
Huawei Technologies
Hansaallee 205,
40549 Dusseldorf,
Germany
Email: Georgios.Karagiannis@huawei.com

Qiong Sun
China Telecom
No.118 Xizhimennei street, Xicheng District
Beijing 100035
P.R. China
Email: sunqiong@ctbri.com.cn

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, Sur-3 building, 3rd floor
Madrid 28050
Spain
Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://people.tid.es/LuisM.Contreras/

Parviz Yegani
JUNIPER NETWORKS
1133 Innovation Way
Sunnyvale, CA 94089
Email: pyegani@juniper.net

John Strassner
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95138 USA
Email: john.sc.strassner@huawei.com

Jun Bi
Tsinghua University
Network Research Center, Tsinghua University
Beijing 100084
China
EMail: junbi@tsinghua.edu.cn

Network Working Group
Internet Draft
Intended status: Standard Track
Expires: November 2015

D. Zhang

A. Zaalouk
EICT

Y. Cheng, Ed
China Unicom

J. Lindblad
Tail-F

M. Klyus
NetCracker

May 11, 2015

VPN Service Management YANG Data Model
draft-zaalouk-supa-vpn-service-management-model-05

Abstract

Currently new services create new opportunities for both network providers and service providers. Simplified Use of Policy Abstractions (SUPA) was proposed to develop a model that abstracts network resources and services and a methodology by which the management and monitoring of network services can be done using standardized policy rules. This document defines a VPN service management yang data model and gives an example for DDC use case.

Status of this Memo

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

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

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

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

Table of Contents

1. Introduction

Currently new services bring new challenges and opportunities for both network and service providers. Meanwhile, legacy services

such as VPN [RFC4110] also need specialized management and controlling capability from the network management systems to improve the experiences for fast deployment and dynamic configuration.

Simplified Use of Policy Abstractions (SUPA) [SUPA-problem-statement] [SUPA-framework] was proposed to introduce the concepts of multi-level and multi-technology network abstractions to address the current separation between development and deployment operations. The first example that SUPA will focus on will be VPN management.

This document introduces YANG [RFC6020] [RFC6021] data models for SUPA configuration. Such models can facilitate the standardization for the interface of SUPA, as they are compatible to a variety of protocols such as NETCONF [RFC6241] and [RESTCONF]. Please note that in the context of SUPA, the term "application" refers to an operational and management applications employed, and possibly implemented, by an operator. The first configuration model is based on the first example - VPN management.

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119]. In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying [RFC2119] significance.

3. Network Service Modules

In this section, several specific network service models are described based on a set of specific network services and the framework of SUPA [SUPA-framework].

3.1. Generic VPN Service Module

A virtual private network (VPN) extends a private network across a public network, such as the Internet. It enables a computer or network-enabled device to send and receive data across shared or public networks as if it were directly connected to the private

network, while benefiting from the functionality, security and management policies of the public network. [VPN]

VPN systems may be classified by multiple ways, e.g., tunnelling protocols, tunnel's termination point location, etc. A typical one among these is by the OSI layer they present to the connecting network, such as Layer 2 circuits or Layer 3 network connectivity.

In use cases of [SUPA-DDC], the links between DCs are VPNs, including L2VPN, L3VPN, etc. In this draft, before going deep into specific VPN services, a generic VPN model is firstly proposed below. It could be used by other specific VPN service models, such as L3VPN service models in [l3vpn-service-yang] and section 3.2 of this draft.

Business VPN usually refers to the order information, while Service VPN usually contains the user's CE information, these two are highly abstraction of VPN service model. From inheritance model perspective, Generic VPN is a base class of all VPN models. L2VPN, L3VPN, Service VPN, Business VPN may be inherited from this base class with corresponding extensions. The Network Manager/Controller [SUPA-framework] can provide three kinds of north interfaces: VPN business models based on the Generic VPN, L2VPN, L3VPN, respectively. This suggests that the Network Manager/Controller can directly deploy L2VPN/ L3VPN business, or VPN service without being informed about the specific VPN type. In other word, the Network Manager/Controller selects specific VPN solutions based on Generic VPN and the specific deployment environment/requirement.

```
module: ietf-supa-vpn
   +--rw vpn-instance
      +--rw vpn-instance* [instance-name]
         +--rw name                              string
         +--rw connection-type?                  enumeration
         +--rw service-type?                     enumeration
         +--rw access-management
         |  +--rw user-name        string
         |  +--rw user-password    string
         +--rw instance-oper-state?              enumeration
         +--rw instance-admin-state?             enumeration
         +--rw life-cycle-state?                 enumeration
         +--rw instance-availability-status?     enumeration
         +--rw instance-cfg-revision?            string
         +--rw instance-sla-policy?              string
         +--rw access-interface* [name]
            +--rw name                              if:interface-ref
            +--rw role                              enumeration
```

There are some candidate attributes listed below under discussion and maybe some of them will be added later:

```
         +--rw network-reliability-parameter
         +--rw performance-management-parameter
         |  +--rw pm-enable?               boolean
         +--rw fault-management-parameter
         |  +--rw alarm-enable?            boolean
         |  +--rw alarm-severity?          inet:uri
```

3.1.1. VPN YANG Model

```
<CODE BEGINS>
module ietf-supa-vpn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-supa-vpn";
  // replace with IANA namespace when assigned
  prefix vpn;

  import ietf-interfaces {
    prefix "if";
  }

  organization "IETF";
  contact
    "Editor: Dacheng Zhang
    dacheng.zdc@alibaba-inc.com

    Ying Cheng
    chengying10@chinaunicom.cn
    ";

  description
    "This YANG module defines a generic VPN
    (Virtual Private Network service)
    configuration model common across all of the
    vendor implementations. ";

  revision 2015-03-24 {
    description
      "Initial revision.";
    reference " RFC 4664, RFC4364, RFC7277";
  }

  container vpn-instance {
    list vpn-instance {
      key " name";
      description
        "Indicates the name of the VPN instance created.";

      leaf name {
        type string;
        description " instance name";
      }

      leaf connection-type {
```

```
            type enumeration {
              enum L2VPN {
                value 0;
                description "L2VPN";
              }
              enum L3VPN {
                value 1;
                description "L3VPN";
              }
            }
            description
              "Indicates the type of VPN, may be L2VPN or L3VPN";
          }

          leaf service-type {
            type enumeration {
              enum full-mesh {
                value "0";
                description "full-mesh";
              }
              enum hub-spoke {
                value "1";
                description "hub-spoke";
              }
            }
            default "full-mesh";
            description "Topology type";
          }

          container access-management{
            leaf user-name {
              type string;
              mandatory true;
              description "User name for this access interface";
            }

            leaf user-password {
              type string;
              mandatory true;
              description
                "User password for the access interface. User
                name and password are listed here because VPN
                need the authentication, one typical way is to
                use user name and password, so that in CE dynamic
                migration case, CEs are able to access with
                authentication regardless location.";
            }
```

```
          description
            "This part gives a typical example for access
            management";
        }

        leaf instance-oper-state {
          type enumeration {
            enum enabled {
              value 1;
              description "VPN Service is enabled";
            }
            enum disabled {
              value 2;
              description "VPN Service is disabled";
            }
          }
          description
            "This part indicates the operational state
            of VPN, enabled or disabled.
            User may use this value to inquire the status
            of this instance";
        }

        leaf instance-admin-state {
          type enumeration {
            enum Enabled {
              value 1;
              description "VPN Service Administratively Enabled";
            }
            enum Disabled {
              value 2;
              description "VPN Service Administratively Disabled";
            }
          }
          description
            "This part indicates the administrative state
            of VPN, e.g., enabled, disabled.
            User may use this value to manage the instance";
        }

        leaf life-cycle-state {
          type enumeration {
            enum planned {
              value 1;
              description "VPN Service Planned";
            }
            enum installed {
```

```
            value 2;
            description "VPN Service Installed";
          }
          enum removed {
            value 3;
            description "VPN Service Removed";
          }
        }
        description
          "This part indicates the life cycle state
          of VPN, e.g., installed, removed, planned.";
      }

      leaf instance-availability-status {
        type enumeration {
          enum degraded {
            value 1;
            description
              "VPN Service Degraded";
          }
          enum failed {
            value 2;
            description
             "VPN Service Failed";
          }
        }
        description
          "This part indicates the availability status
          of VPN, e.g., degraded, failed.";
      }

      leaf instance-cfg-revision {
        type string;
        description
          "Indicates current service configuration revision";
      }

      leaf instance-sla-policy {
        type string;
        description
          "Indicates SLA decision policy name";
      }

      list access-interface {
        key name;
        description "Access interface name.";
```

```
            leaf name {
              type if:interface-ref;
              description "Access interface name.";
            }

            leaf role {
              when "../../service-type = 'full-mesh'" {
                description
                  "The role is always center-if in a
                   hub-spoke topology, in a full-mesh
                   topology, the role needs to be
                   specified as center-if or edge-if.";
              }
              type enumeration {
                enum edge-if {
                  value 0;
                  description "Edge interface";
                }
                enum center-if {
                  value 1;
                  description "Center interface";
                }
              }
              mandatory true;
              description
                "An edge interface (edge-if) indicates... FIXME";
            }
          }
        }
        description
        "Generic VPN is a base class for VPN models.
        Other VPN models may be inherited from this base
        class with corresponding extensions.";
      }
    }
    <CODE ENDS>
```

3.2. L3VPN Service Module

A Layer 3 Virtual Private Network (L3VPN) interconnects sets of hosts and routers based on Layer 3 addresses and forwarding. L3VPN can be based on MPLS or IP technologies. L3VPN is a PE-based VPN managed by operators. L3VPN is widely used in carrier metro networks to provide VPN service for enterprise users.

A L3VPN model is a collection of L3VPN instances. A L3VPN instance contains a set of access interfaces to network devices as well as other attributes, such as routing protocol, address family, topology, and so on.

To configure a L3VPN instance, the administrator needs to specify which port(s) of a network device belongs to a L3VPN instance. Those ports and network device information can be derived from a network topology model in a network management system. The administrator also needs to specify what routing protocol needs to be configured for a L3VPN instance.

The following describes a abstracted L3VPN information model for DDC service to use, based on which users can develop applications to configure L3VPN instances for DDC service. L3SM is also working on an elaborate L3VPN service model. After it’s done DDC service in SUPA may directly refer the output of L3SM. The abstracted L3VPN model here is a subset of the one of L3SM.

```
module: ietf-supa-abstracted-l3vpn
   +--rw l3vpns
      +--rw l3vpn-instance* [name]
         +--rw name                              string
         +--rw service-type?                     enumeration
         +--rw address-family?                   enumeration
         +--rw instance-oper-state?              enumeration
         +--rw instance-admin-state?             enumeration
         +--rw life-cycle-state?                 enumeration
         +--rw instance-availability-status?     enumeration
         +--rw instance-cfg-revision?            string
         +--rw instance-sla-policy?              string
         +--rw service-quality?                  enumeration
         +--rw access-interface* [name]
         |  +--rw name       if:interface-ref
         |  +--rw address    inet:ip-prefix
         |  +--rw role       enumeration
         +--rw user-name                         string
         +--rw user-password                     string
         +--rw physical-node-id                  string
         +--rw physical-access-interface         if:interface-ref
         +--rw routing
            +--rw protocol?        enumeration
            +--rw bgp-attribute
            |  +--rw remote-as-number?      inet:as-number
            |  +--rw remote-peer-address    inet:ip-address
            +--rw igp-attribute
               +--rw protocol-id?   uint32
```

3.2.1. L3VPN YANG Model

```
<CODE BEGINS>
module ietf-supa-abstracted-l3vpn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-supa-abstracted-
l3vpn";
  // replace with IANA namespace when assigned
  prefix abstracted-l3vpn;

  import ietf-inet-types {
    prefix inet;
  }
  import ietf-interfaces {
    prefix if;
  }

  organization "IETF";
  contact
```

```
       "Editor: Dacheng Zhang
        dacheng.zdc@alibaba-inc.com

        Adel Zaalouk
        adel.ietf@gmail.com

        Kostas Pentikousis
        k.pentikousis@eict.de

        Jan Lindblad
        janl@tail-f.com

        Maxim Klyus
        klyus@NetCracker.com
       ";

     description
       "This YANG module an abstracted L3VPN information
        model for DDC service to useL3VPN, based on which
        users can develop applications to configure L3VPN
        instances for DDC service.

        Terms and Acronyms
          L3VPN: Layer 3 Virtual Private Network
       ";

     revision 2015-05-04 {
       description "Initial revision.";
       reference "RFC4364, RFC7277";
     }

     container l3vpns {
       description "an abstracted L3VPN information
        model for DDC service to use L3VPN.";

       list l3vpn-instance {
         key name;
         description
           "Indicates the name of the VPN instance.";

         leaf name {
           type string;
           description "L3VPN instance name.";
         }

         leaf service-type {
           type enumeration {
```

```
            enum full-mesh {
              value 0;
              description "Full-mesh topology.";
            }
            enum hub-spoke {
              value 1;
              description "Hub-spoke topology.";
            }
          }
          default "full-mesh";
          description "Topology type.";
        }

        leaf address-family {
          type enumeration {
            enum ipv4uni {
              value 0;
              description "ipv4 unicast address family.";
            }
            enum ipv6uni {
              value 1;
              description "ipv6 unicast address family.";
            }
          }
          default "ipv4uni";
          description "Address family type: IPv4 or IPv6.";
        }

        leaf instance-oper-state {
          type enumeration {
            enum enabled {
              value 1;
              description "VPN Service is enabled";
            }
            enum disabled {
              value 2;
              description "VPN Service is disabled";
            }
          }
          description
            "This part indicates the operational state
            of VPN, enabled or disabled.
            User may use this value to inquire the status
            of this instance";
        }

        leaf instance-admin-state {
```

```
            type enumeration {
              enum Enabled {
                value 1;
                description "VPN Service Administratively Enabled";
              }
              enum Disabled {
                value 2;
                description "VPN Service Administratively Disabled";
              }
            }
            description
              "This part indicates the administrative state
              of VPN, e.g., enabled, disabled.
              User may use this value to manage the instance";
          }

          leaf life-cycle-state {
            type enumeration {
              enum planned {
                value 1;
                description "VPN Service Planned";
              }
              enum installed {
                value 2;
                description "VPN Service Installed";
              }
              enum removed {
                value 3;
                description "VPN Service Removed";
              }
            }
            description
              "This part indicates the life cycle state
              of VPN, e.g., installed, removed, planned.";
          }

          leaf instance-availability-status {
            type enumeration {
              enum degraded {
                value 1;
                description
                  "VPN Service Degraded";
              }
              enum failed {
                value 2;
                description
                 "VPN Service Failed";
```

```
            }
          }
          description
            "This part indicates the availability status
            of VPN, e.g., degraded, failed.";
        }

        leaf instance-cfg-revision {
          type string;
          description
            "Indicates current service configuration revision";
        }

        leaf instance-sla-policy {
          type string;
          description
            "Indicates SLA decision policy name";
        }

        leaf service-quality {
          type enumeration {
            enum gold {
              value 0;
              description
               "Best quality of service is guaranteed.";
            }
            enum silver {
              value 1;
              description
               "Better quality of service than bronze class
               is guaranteed.";
            }
            enum bronze {
              value 2;
              description
              "Average quality of service is guaranteed.";
            }
          }
          default "bronze";
          description "Quality of service.";
        }

        list access-interface {
          key name;
          description "Access interfaces.";

          leaf name {
```

```
            type if:interface-ref;
            description "Access interface name.";
          }

          leaf address {
            type inet:ip-prefix;
            mandatory true;
            description "Access interface address, IPv4 or IPv6.";
          }

          leaf role {
            when "../../service-type = 'full-mesh'" {
              description
                "The role is always center-if in a
          hub-spoke topology, in a full-mesh
          topology, the role needs to be
          specified as center-if or edge-if.";
            }
            type enumeration {
              enum edge-if {
                value 0;
                description "Edge interface";
              }
              enum center-if {
                value 1;
                description "Center interface";
              }
            }
            mandatory true;
            description
        "center-if is only available in hub-spoke
        mode; There are two scenarios: in full
        mesh mode, the role of all the access-interface
        is edge-if, while in hub-spoke mode, the role of
        the interface of hub node is ceter-if, the one
              of spoke node is edge-if";
          }
        }

        leaf user-name {
          type string;
          mandatory true;
          description "User name for this access interface";
        }

        leaf user-password {
          type string;
```

```
          mandatory true;
          description
        "User password for the access interface in
        encrypted format. User name and password are
        listed here because VPN need the
        authentication, one typical way is to use
            user name and password, so that in CE dynamic
        migration case, CEs are able to access with
        authentication regardless location.";
        }

        leaf physical-node-id {
          type string;
          mandatory true;
          description "This is the physical node ID
          of access PE";
        }

        leaf physical-access-interface {
          type if:interface-ref;
          mandatory true;
          description " This is the Physical access
          interface of access PE.";
        }

        container routing {
          description
            "Routing configuration between PE and CE.";

          leaf protocol {
            type enumeration {
              enum bgp {
                value 0;
                description "bgp routing between PE and CE";
              }
              enum ospf {
                value 1;
                description "ospf routing between PE and CE";
              }
              enum isis {
                value 2;
                description "isis routing between PE and CE";
              }
              enum rip {
                value 3;
                description "rip routing between PE and CE";
              }
```

```
                enum static {
                  value 4;
                  description "static routing between PE and CE";
                }
              }
              default "ospf";
              description "Routing protocol between PE and CE.";
            }

            container bgp-attribute {
              when "../protocol = 'bgp'" {
                description "BGP specific parameters.";
              }

              leaf remote-as-number {
                type inet:as-number;
                description "Remote BGP peer as-number.";
              }

              leaf remote-peer-address {
                type inet:ip-address;
                mandatory true;
                description "Remote BGP peer address.";
              }
              description "Detailed configuration for BGP.";
            }

            container igp-attribute {
              when "../protocol = 'ospf' or
                    ../protocol = 'isis' or
                    ../protocol = 'rip'  or
                    ../protocol = 'static'" {
                description "IGP specific parameters.";
              }
              leaf protocol-id {
                type uint32;
                default 0;
                description "Valid only when protocol is IGP;
                             it can be AS number.";
              }
              description " This part defines the attribute of IGP.";
            }
          }
        }
      }
    }
    <CODE ENDS>
```

3.3. L2VPN Service Module

This section describes service model for Ethernet L2VPN.

There are different ways of classifying L2VPNs. According to the Ethernet services defined by Metro Ethernet Forum, there are mainly three types of Ethernet L2VPN service that can be provided by service providers: E-line, E-tree and E-lan.

- o E-line is point to point service.
- o E-lan is multipoint to multipoint service.
- o E-tree is multipoint to multipoint service, but the communications between some consumer sites are not allowed.

Meanwhile according to [RFC4664], there are two fundamentally different kinds of Layer 2 VPN service that a service provider could offer to a customer: Virtual Private Wire Service (VPWS) and Virtual Private LAN Service (VPLS).

- o VPWS is a L2 VPN service that provides L2 point-to-point service.
- o VPLS is a L2 VPN service that emulates LAN service across a Wide Area Network (WAN).

Based on different degrees of abstraction, the interfaces can be categorized into two kinds: a) The user can provide service type and site information to the controller and controller create VPN automatically, based on the network, maybe by pseudo wire or other method. This is the more abstracted Ethernet L2VPN service interface. b) If the provider is MPLS or MPLS-TP, controller can provide less abstracted interfaces to the user. The user can use such interfaces to control the network more agility. This document describes the former one, a more abstracted model for L2VPN.

The following describes the information model for L2VPN, based on which users can develop applications to configure L2VPN instances.

```
module: ietf-supa-l2vpn
   +--rw l2vpn-instances
      +--rw l2vpn-instance* [name]
         +--rw name                              string
         +--rw service-type                      enumeration
         +--rw instance-oper-state?              enumeration
         +--rw instance-admin-state?             enumeration
         +--rw life-cycle-state?                 enumeration
         +--rw instance-availability-status?     enumeration
         +--rw instance-cfg-revision?            string
         +--rw instance-sla-policy?              string
         +--rw service-quality?                  enumeration
         +--rw access-interface* [interface-id]
            +--rw interface-id           if:interface-ref
            +--rw role                   enumeration
            +--rw node-id                string
            +--rw interface-name         if:interface-ref
            +--rw access-type?           enumeration
            +--rw user-circuit-bitmap    string
```

3.3.1. L2VPN YANG Model

```
<CODE BEGINS>
module ietf-supa-l2vpn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-supa-l2vpn";
  // replace with IANA namespace when assigned
  prefix l2vpn;

  import ietf-interfaces {
    prefix if;
  }

  organization "IETF";
  contact
    "Editor: Ying Cheng
    chengying10@chinaunicom.cn

    Vikram Choudhary
    vikschw@gmail.com
    ";

  description
    "This YANG module defines a generic service
    configuration model for L2VPN (Layer 2 Virtual
    Private Network), based on which programmers can
    develop applications to configure L2VPN instances.";
```

```
     revision 2015-04-09 {
       description
         "Initial revision";
       reference "RFC4664";
     }

     container l2vpn-instances {
       description "an abstracted L3VPN information
        model for DDC service to use L3VPN.";

       list l2vpn-instance {
         key "name";
         description
           "Indicates the name of the L2VPN instance";

         leaf name {
           type string;
           description "L2VPN instance name";
         }

         leaf service-type {
           type enumeration {
             enum e-line {
               value "0";
               description "the service type is e-line";
             }
             enum e-tree {
               value "1";
               description "the service type is e-tree";
             }
             enum e-lan {
               value "2";
               description "the service type is e-lan";
             }
           }
           mandatory true;
           description
             "The service type of this instance,
             user may choose from e-line, e-tree, e-lan";
          }

         leaf instance-oper-state {
           type enumeration {
             enum enabled {
               value 1;
               description "VPN Service is enabled";
```

```
            }
            enum disabled {
              value 2;
              description "VPN Service is disabled";
            }
          }
          description
            "This part indicates the operational state
            of VPN, enabled or disabled.
            User may use this value to inquire the status
            of this instance";
        }

        leaf instance-admin-state {
          type enumeration {
            enum Enabled {
              value 1;
              description "VPN Service Administratively Enabled";
            }
            enum Disabled {
              value 2;
              description "VPN Service Administratively Disabled";
            }
          }
          description
            "This part indicates the administrative state
            of VPN, e.g., enabled, disabled.
            User may use this value to manage the instance";
        }

        leaf life-cycle-state {
          type enumeration {
            enum planned {
              value 1;
              description "VPN Service Planned";
            }
            enum installed {
              value 2;
              description "VPN Service Installed";
            }
            enum removed {
              value 3;
              description "VPN Service Removed";
            }
          }
          description
            "This part indicates the life cycle state
```

```
            of VPN, e.g., installed, removed, planned.";
          }

          leaf instance-availability-status {
            type enumeration {
              enum degraded {
                value 1;
                description
                  "VPN Service Degraded";
              }
              enum failed {
                value 2;
                description
                 "VPN Service Failed";
              }
            }
            description
              "This part indicates the availability status
              of VPN, e.g., degraded, failed.";
          }

          leaf instance-cfg-revision {
            type string;
            description
              "Indicates current service configuration revision";
          }

          leaf instance-sla-policy {
            type string;
            description
              "Indicates SLA decision policy name";
          }

          leaf service-quality {
            type enumeration {
              enum gold {
                value 0;
                description
                 "Best quality of service is guaranteed.";
              }
              enum silver {
                value 1;
                description
                 "Better quality of service than bronze class
                 is guaranteed.";
              }
              enum bronze {
```

```
            value 2;
            description
            "Average quality of service is guaranteed.";
          }
        }
        default "bronze";
        description "Quality of service.";
      }

      list "access-interface" {
        key "interface-id";
        description "Access interface ID";

        leaf interface-id {
          type if:interface-ref;
          description "Access interface ID";
        }

        leaf role {
          when "../../instance-service-type = 'e-tree'" {
        description
        "root or leaf, only available in e-tree mode";
      }
          type enumeration {
            enum root {
              value "0";
              description "root interface";
              }
            enum leaf {
              value "1";
              description "leaf interface";
            }
          }
          mandatory true;
          description
          "the role of interface";
        }

        leaf node-id {
          type string;
          mandatory true;
          description "physical node ID of access PE";
        }

        leaf interface-name {
          type if:interface-ref;
          mandatory true;
```

```
            description
              "physical access interface name of access PE";
          }

          leaf access-type {
            type enumeration {
              enum port{
                value "0";
                description "The access type is port.";
              }
              enum vlan{
                value "1";
                description "The access type is vlan.";
              }
        }
              default "port";
              description
                "This value describes the access type,
                options are port and vlan; port is chose
                when the access is the entire port, while vlan
                means the access is the vlan of the port.";
          }

          leaf user-circuit-bitmap {
            when "../../ access-type = 'vlan'"{
          description "Only available when access-type is vlan.";
        }
              type string;
              mandatory true;
              description
                "This value describes the scope of vlan:
        use ',' or '-' to indicate the scope, e.g. 1,3,5-7.";
            }
          }
        }
      }
    }
   }
   <CODE ENDS>
```

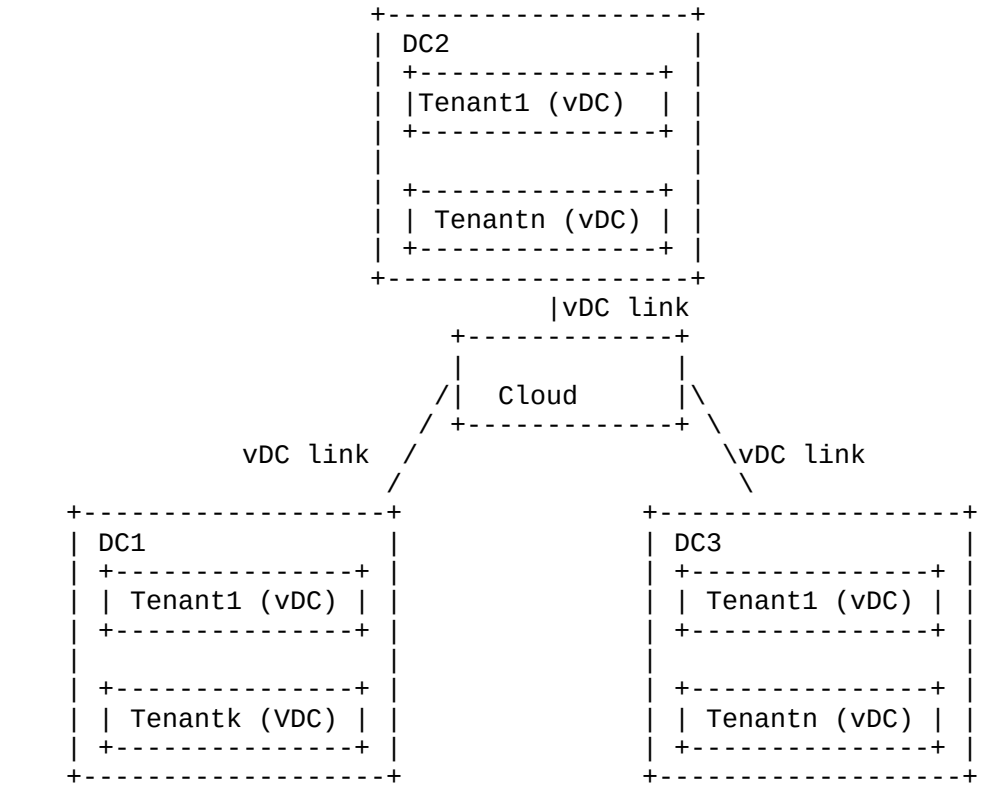
3.4. Module for DDC services

The following describes SUPA VPN management model designed for DDC services use case [SUPA-DDC]. [SUPA-DDC] took a large-scale Internet Data Center (IDC) operator as an example to describe what SUPA needs to do including DDC service initiation, VPN-based

connectivity initiation, optimize traffic route, traffic adjustment and monitor.

Module "ietf-supa-ddc" defines generic VPN management aspects which are common to all DDC services use case regardless of their type of vendor. In effect, the module can be viewed as providing a generic VPN management for DDC services.

This model is designed based on vDC Connectivity use case in section 6.2 of [SUPA-DDC]. The figure below showed the tenants distributed on multiple DCs and the connectivity among them. Service data model and policy data model can be defined to automate or simplify the links configuration for vDCs.



Please note that, for DDC model, it is essentially a data center service model, but with the help of L3VPN do bearing, so it refers to L3VPN business model (not inheritance relationships).

```
module: ietf-supa-ddc
   +--rw ddc-services
      +--rw ddc-service* [name]
         +--rw name               string
         +--rw tenant-name?       string
         +--rw dc-name*           string
         +--rw interface-name*    string
         +--rw connection-type?   enumeration
         +--rw connection-name    leafref
         +--rw bandwidth?         uint32
         +--rw latency?           uint32
```

3.4.1. Model for DDC services

```
<CODE BEGINS>
module ietf-supa-ddc {
  namespace "urn:ietf:params:xml:ns:yang:ietf-supa-ddc";
  // replace with IANA namespace when assigned
  prefix ddc;

  // import ietf-supa-vpn,ietf-supa-abstacted-l3vpn,
  // ietf-supa-l2vpn



  organization "IETF";
  contact
     "Editor: Ying Cheng
      chengying10@chinaunicom.cn

      Maxim Klyus
      klyus@NetCracker.com
     ";

  description
     "This YANG module defines a component that describing
      the ddc service model for creating and optimizing
      tenant's DC (data center) services that are deployed
      in multiple data centers.

      Terms and Acronyms
        DDC: Distributed Data Center
        L2VPN: Layer 2 Virtual Private Network
        L3VPN: Layer 3 Virtual Private Network
      ";
```

```
     revision 2014-12-25 {
        description
           "Initial revision.";
        reference "RFC4364, RFC7277";
     }

     container ddc-services {
       description
         "To create service for tenant's network that are deployed
         in multiple data centers. The following data are needed:
         name of data centers that the tenant's service are
         deployed in, connected method between data centers for
         the tenant (e.g. L2VPN, l3VPN, Native IP, etc.), name
         of tenant, ID of networks that belong to the tenant";

       list ddc-service {
         key "name";
         description
           "Overall ddc operational data, including the names of
           data center,the connection method between data centers,
           name of service, etc.";

         leaf name {
           type string;
           description
             "Indicates the name of the service";
         }

         leaf tenant-name {
           type string;
           description
             "Indicates the name of the tenant for whom
         the ddc service is being created.";
         }

         leaf-list dc-name {
           type string;
           description
             "List of the names of data center on which
             tenant's service is deployed in.";
         }

         leaf-list interface-name {
           type string;
           description
             "Indicates a set of access interface
             names of the network device that the
```

```
            data centers (deployment of tenant's
            service)are connected to.";
          }

          leaf connection-type {
            type enumeration {
              enum L2VPN {
                value 0;
                description "L2VPN";
              }
            enum L3VPN {
                value 1;
                description "L3VPN";
              }
              enum native-ipv4 {
                value 2;
                description "native IPv4";
              }
              enum native-ipv6 {
                value 3;
                description "native IPv6";
              }
            }
            description
              "Indicates the connection  type between the
              Data centers on which tenant service is being
              deployed . The connection type may be VPN
              (L2VPN or L3VPN) or Native IP (IPv4 or IPv6)";
          }

          leaf connection-name {
        type leafref { path "/l2vpn:l2vpn-instance/
          instance-name"; }
            mandatory true;
            description
              "Indicates the name of the connection e.g.,VPN
              instance";
          }

          leaf bandwidth {
            type uint32;
            description
              "Indicates the bandwidth of the network connection
              instance that is created for tenant.";
          }

          leaf latency {
```

```
            type uint32;
            description
              "Indicates the latency of the network connection
              instance that is created for tenant.";
          }
        }
      }
    }
    <CODE ENDS>
```

4. Annex A (Service instance common attributes)

The below code with set of common attributes can be implemented for each type of VPN service.

Attributes description:

Operational State - operability of a VPN service is described by the instance-oper-state read-only attribute, which has two possible values: disabled and enabled. Operational State changes can be driven by specific events associated with the VPN components and specific administrative VPN transitions.

instance-oper-state attribute states:
- enabled: service is enabled and can operate normally or with some errors
- disabled: service disabled administratively or automatically based on some critical errors

Administrative State - administration of managed VPN service described by instance-admin-state read-write attribute, which has two possible values: disabled and enabled. This attribute operates independently of the operability and usage of managed VPN and could be changed automatically based on specific events associated with VPN components or manually.

instance-admin-state attribute states:
- enabled: service administratively enabled, service configuration structure unchanged, but service component states can be changed)
- disabled: service administratively disabled, service configuration structure unchanged, but service component states can be changed)

Life Cycle State - planning and tracking of managed VPN service described by life-cycle-state read-write attribute. Inventoried VPN

components also may have a life cycle attribute so that their deployment can be planned, tracked and managed.

The life-cycle-state attribute shall have one of the following values
- planned: VPN is planned; configuration is ready but not installed in the network.
- installed: VPN service is ready for production; configuration is installed in the network.
- removed: The VPN service has been removed, configuration uninstalled from the network and saved for the history.

Availability Status - availability of a VPN service described by the instance-availability-status read-only attribute, which can be zero or has one of two possible values: degraded or failed. Availability Status changes can be driven by specific events associated with the VPN components and specific administrative VPN transitions.

instance-availability-status attribute states:
- degraded: VPN is still can operate but some of the VPN components have an errors.
- failed: VPN service can’t operate properly; because some of the VPN components have critical errors and VPN Operational State seems to be disabled.

Configuration Revision - configuration management of a VPN service is described by the instance-cfg-revision read-write string attribute. Service configuration could be modified (expanded/reduced) and VPN service instance will be unchanged but service components can be changed. Based on this attribute configuration changes can be tracked for service modification history purpose and last active configuration can be obtained.

SLA Policy - SLA policy reference described by the instance-sla-policy read-write string attribute. Decision about VPN service Operational State (Enabled / Disabled) and Availability State (Degraded / Failed) can be very complicated and each SP can have own decision process and internal service integrity check implementation. SLA Policy attribute SP can implement some SLA decision policy per VPN group service basis (when different VPNs have similar SLA policy).

```
   <CODE BEGINS>

   leaf instance-oper-state {
     type enumeration {
       enum enabled {
         value 1;
         description
           "VPN Service is Enabled";
       }
       enum disabled {
         value 2;
         description
           "VPN Service is Disabled";
       }
     }
   }

   leaf instance-admin-state {
     type enumeration {
       enum enabled {
         value 1;
         description
           "VPN Service Administratively Enabled";
       }
       enum disabled {
         value 2;
         description
         "VPN Service Administratively Disabled";
       }
     }
   }

   leaf life-cycle-state {
     type enumeration {
       enum planned {
         value 1;
         description
           "VPN Service Planned";
       }
       enum installed {
         value 2;
         description
           "VPN Service Installed";
       }
       enum removed {
         value 3;
         description
```

```
            "VPN Service Removed";
        }
      }
    }

    leaf instance-availability-status {
      type enumeration {
        enum degraded {
          value 1;
          description
            "VPN Service Degraded";
        }
        enum failed {
          value 2;
          description
            "VPN Service Failed";
        }
      }
    }

    leaf instance-cfg-revision {
      type string;
      description
          "Indicates current service configuration revision";
    }

    leaf instance-sla-policy {
      type string;
      description
        "Indicates SLA decision policy name";
    }

    <CODE ENDS>
```

5. Security Considerations

TBD

6. IANA Considerations

This document has no actions for IANA.

7. Contributors and Acknowledgments

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in alphabetical order: Feng Dong, Jing Huang, Kepeng Li, Junru Lin, Felix Lu, Wu Nan, Juergen Schoenwaelder, Yiyong Zha, and Cathy Zhou.

Contributors:

Andy Bierman
andy@yumaworks.com

Juergen Schoenwaelder
j.schoenwaelder@jacobs-university.de

8. Additional Author List

The following are extended authors who contributed to the effort:

Kostas Pentikousis
EICT GmbH
Torgauer Strasse 12-15
Berlin 10829
Germany
Email: k.pentikousis@eict.de

Vikram Choudhary
Huawei
Email: vikram.choudhary@huawei.com

Will Liu
Huawei
Email: liushucheng@huawei.com

9. References

9.1. Normative References

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

[RFC6020] Bjorklund, M., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6021] Schoenwaelder, J., "Common YANG Data Types", RFC 6021, October 2010.

[RFC4110] Callon, R. and M. Suzuki, "A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs)", RFC 4110, July 2005.

[RFC3272] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and X. Xiao, "Overview and Principles of Internet Traffic Engineering", RFC 3272, May 2002.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, December 1998.

[RFC4664] L. Andersson and E. Rosen, "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, September 2006.

9.2. Informative References

[SUPA-framework] C. Zhou, L. M. Contreras, Q. Sun, and P. Yegani, " The Framework of Simplified Use of Policy Abstractions (SUPA) ", IETF Internet draft, draft-zhou-supa-framework, February 2015.

[SUPA-problem-statement] G. Karagiannis, Q. Sun, Luis M. Contreras, P. Yegani, JF Tremblay and J. Bi, "Problem Statement for Simplified Use of Policy Abstractions (SUPA)", IETF Internet draft, draft-karagiannis-supa-problem-statement, January 2015.

[SUPA-DDC] Y. Cheng, and JF. Tremblay, "Use Cases for Distributed Data Center Applications in SUPA", IETF Internet draft, draft-cheng-supa-ddc-use-cases, January 2015

[RESTCONF] Bierman, A., Bjorklund, M., Watsen, K., and R. Fernando, "RESTCONF Protocol", draft-ietf-netconf-restconf (work in progress), July 2014.

[VPN] Mason, Andrew G. (2002). Cisco Secure Virtual Private Network. Cisco Press. p. 7.

[l3vpn-service-yang] S. Litkowski, R. Shakir, L. Tomotaki and K. D'Souza, "YANG Data Model for L3VPN service delivery", IETF Internet draft, draft-l3vpn-service-yang, February 2015.

Authors' Addresses

Dacheng Zhang

Chaoyang Dist
Beijing 100000
P.R. China
Dacheng.zhang@gmail.com

Adel Zaalouk
EICT GmbH
Torgauer Strasse 12-15
Berlin 10829
Germany
Email: adel.ietf@gmail.com

Ying Cheng (Editor)
China Unicom
P.R. China

Email: chengying10@chinaunicom.cn

Jan Lindblad
Tail-F

Email: janl@tail-f.com

Maxim Klyus
NetCracker

Email: klyus@NetCracker.com

Network Working Group
Internet-Draft

Intended status: Informational
Expires: November 08, 2015

C. Zhou
Huawei Technologies
L. M. Contreras
Telefonica
Q. Sun
China Telecom
P. Yegani
Juniper Networks
May 08, 2015

The Framework of Simplified Use of Policy Abstractions (SUPA) draft-zhou-supa-framework-02

Abstract

Currently, there are network services that impose specific demands on a communication network. This document describes the SUPA basic architecture, its elements and interfaces. The main SUPA architecture entities are the Service Management (SM) and the Network Manager/Controller (NM/NC). The SM is a functional entity that creates and runs network services by using a set of NM/NCs. The NM/NC is a functional entity that provides one or more of the following functions: (1) the generation, maintenance and release of network topologies, (2) the generation, maintenance, and release of network service-specific abstractions, (3) the mapping between network service-specific abstractions and the network topology, and (4) the mapping between network service-specific abstractions and network device configuration. Both the SM and the NM/NC may be made up of multiple distinct entities that collectively perform their functions.

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 April 27, 2015.

Copyright Notice

Requirements Language

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

Table of Content

1. Introduction

As the Internet grows, new services are created, new devices are connected, and new users use the Internet. These and other factors contribute to the significant increase in the amount and type of network traffic. This in turn makes network management and configuration more and more complicated.

However, the need for dynamic and real-time configuration changes is required. One example is Inter-Data Center (DC) traffic steering and tunneling, based on real-time network status. Dedicated network management applications are required to automate the complicated and

dynamic network configuration present in today's systems. Such applications need interoperable data models of network topology, network services, and policy rules to support the design, deployment, and maintenance of network services. Providing these data models to network management applications may provide significant improvements in configuration agility, error detection, and uptime for operators.

However, in order to scale and to ensure configuration change consistency, existing network configuration schemes must be simplified through the use of abstraction. This increases the programmatic control over such systems. Simplified models are able to provide a wide range of granularity for various applications and network services needs, from the lower-level physical network to high-level application services.

An abstract view of a network infrastructure can be realized using one or more network topology data models. Each such topology model, whether physical or logical, may be used as its own reusable managed object. This enables existing topologies to be reused to create new topologies. This applies to models of different layers, including the application layer (L7), IP/network layer (L3), and lower layers (L0-L2, such as MPLS, SDH, OTN, WDM). The network resources may include physical and/or virtual network nodes and links.

A network service data model is service specific, and usually relies on a network topology data model. The network service data model defines the behavior of the network service, which is characterized by one or more metrics that include performance, dependability, and security specifications.

One method to automate service configuration is to use a policy data model. Policy, in conjunction with network service and device data models, can be used to tell the Network Manager/Controller (NM/NC) how to generate Network Element (NE) configurations using network service data models in combination with topology data models. For example, this process can be used to choose a path for VPN which will involve a set of NE's.

The main goal of this document is to specify the SUPA reference architecture, its elements, and its interfaces.

YANG data models (e.g., see [RFC6020], [RFC6991]) can be used for representing service and topology models.

2. Terminology

The terminology used in the SUPA problem statement draft [ID.karagiannis-supa-problem-statement] also applies to this draft.

NE	Network Element
NC	Network Controller
NM	Network Manager

NM/NC Network Manager/Controller
SM Service Management

3. SUPA Framework

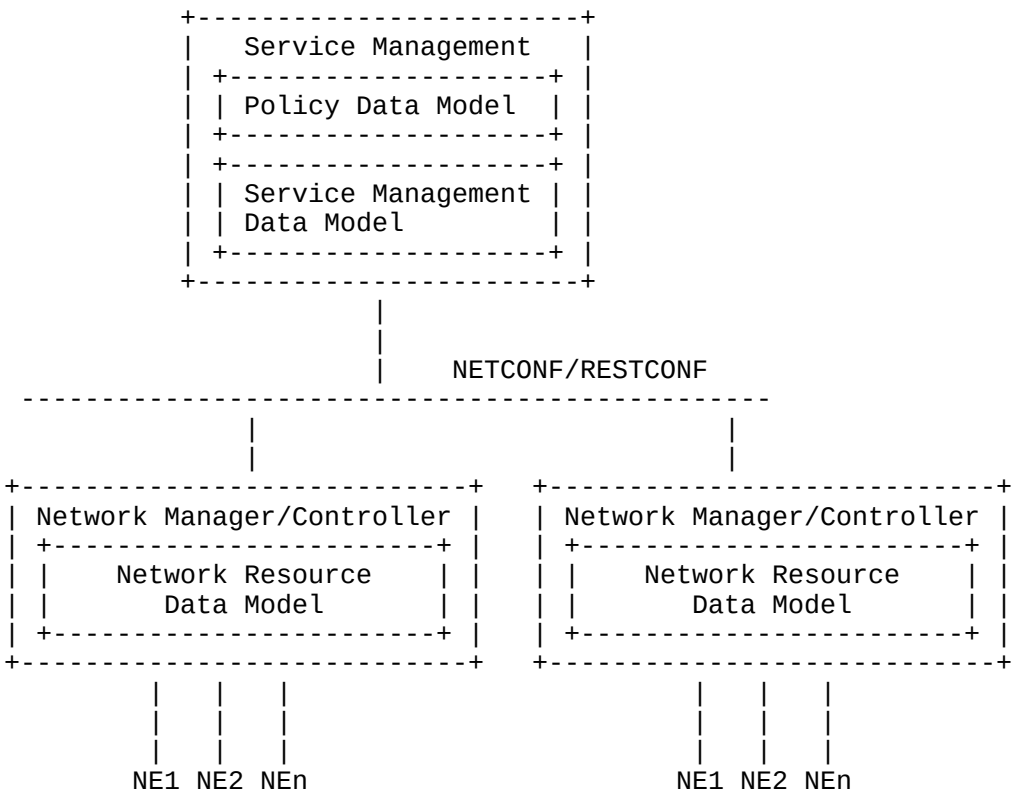


Figure 1 SUPA Framework

An overview of the SUPA framework is shown in Figure 1. The network entities used in this framework are:

SM: Service Management, which represents one or more network entities that are running and controlling network services.

Policy Data Model
Model of policy rules for managing the network service and mapping services dynamically to the network topology and network resources. Policy data models are used to describe high level service requirements, such as routing requirements.

Example of policy data model can be found in [ID.draft-strassner-supa-generic-policy-info-model] and [ID.draft-bi-supa-policy-model].

There can be various types of policies, including service specific policies and network-wide policies. There can be a

centralized entity managing the network-wide policies, which may be called as policy manager. The policy manager can be located in the SM or in a separate location.

Policy data models now considered by SUPA are generic policy data model, ECA (event, condition, action) as described in SUPA charter. The work may be extended in the future.

Service Management Data Model
Model of the network service (e.g., VPNs) and the network resources required by the network service to be correctly deployed and executed on the physical and/or virtual topology.

Example of service management data model can be found in [ID.draft-zaalouk-supa-vpn-service-management-model].

NM/NC: Network Manager / Controller, which 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 (NEs)).

Network Resource Data Model
Model of the 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.

Example of network resource data model can be found in [ID.draft-contreras-supa-yang-network-topo]

SUPA will not define network resource data model, which is out of the scope of SUPA. SUPA will make use of network resource data models defined by other WGs or SDOs.

Network Element (NE), which handles packets based on the network management and controlling procedures. NEs can interact with local or remote NM/NC in order to exchange information, such as configuration information, policy enforcement capabilities, and network status.

Service Management (SM) communicates with Network Manager/Controller (NM/NC) using an appropriate protocol, such as NETCONF [RFC6241] or RESTCONF [ID.draft-ietf-netconf-restconf].

NM/NC exchanges configuration information with NEs and derives the current network topology that contains the NEs, and also the capabilities and status of NEs, which will be stored in network resource data model. NM/NC may also communication with traditional network management system to retrieve the above information. It can use existing network management and signaling protocols, such as I2RS [I2RS], NETCONF [NETCONF], RESTCONF [ID.draft-ietf-netconf-restconf], etc.

Service Management (SM) will send policy data model and service management data model, which will in conjunction with network resource data model be mapped to detail NEs' configurations by network manager / controller.

4. Framework Functional Entities

4.1. Service Management

There are a wide variety of communication services offered by service providers.

The Service Management (SM) is a functional entity, residing at the Application layer, which enables network services, such as:

- o) Network services (e.g., L2VPN, L3VPN, etc.)
- o) application-specific policies

SM will push service management data model and policy data model to NM/NC for service deployment.

4.2. Network Manager/Controller

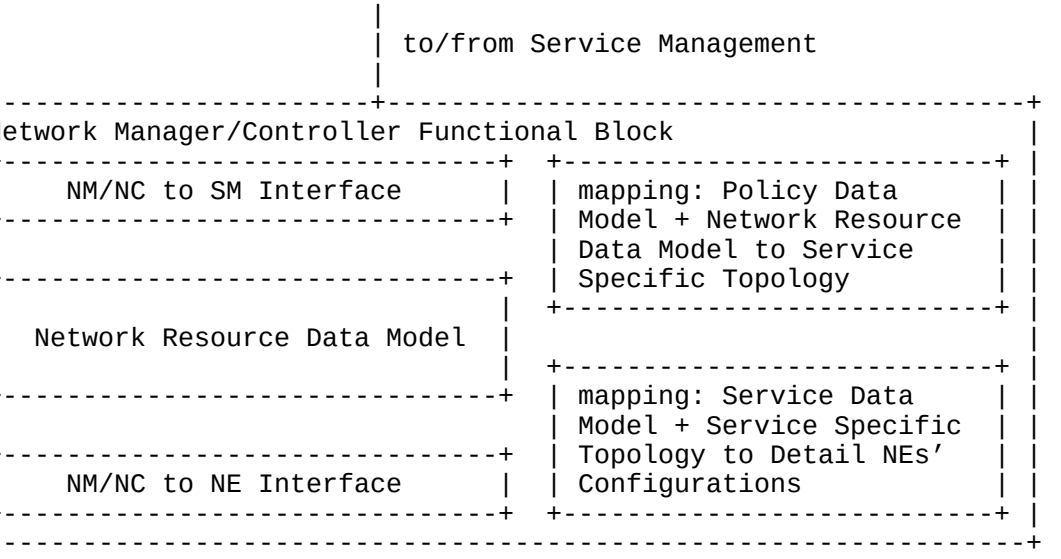


Figure 2 NM/NC Functional Blocks

Network Manager/Controller (NM/NC) is a functional entity that is able to generate and maintain desired and current topologies of the network infrastructure. As part of this process, it is also responsible for reserving and releasing network resources that are required to support network services in a given network infrastructure.

The NM/NC contains a set of data models, functions and APIs, including:

o) Network Resource Data Model
Maintain an up to date topology of the network infrastructure, including capabilities and current status of NEs.

o) Mapping: service specific topology
This mapping procedure will combine the policy data model and service management data model and generate a specific topology.

o) Mapping: target NE(s) detail configurations
This mapping procedure will use the service specific topology generated in the previous procedure and the service management data model to generate detail NE(s) configurations.

o) NM/NC to traditional network management system interface: provides the interface with existing network management system protocols (e.g., I2RS [I2RS], NETCONF, etc.) to request configuration and status information, and push configuration changes to NEs.

o) NM/NC to SM interface: used to support the communication between the SM and NM/NC. The candidate protocols used for this purpose could be either NETCONF [RFC6241] or RESTCONF [ID.draft-ietf-netconf-restconf].

An example of the mapping procedures can be that, a service requires that a link from A to B is created; and the policy requires that the hops of this link should not exceed N. Then when NM/NC receives the policy data model and service management data model from SM, it will first apply the policy data model to the network resource data model and get a sub-set topology which can fulfill the hops limit requirements. Then NM/NC will further generate detail configurations for target NE(s). The mapping procedures can be enforced by functional entity called policy agent.

After the service is deployed, if there is a network topology change, network configurations for this service may need to be updated accordingly. A possible solution is to repeat the mapping procedures, and generate configurations for NEs (maybe another set of NEs). This requires that NM/NC maintains a copy of the service management data models and policy data models.

For more detail about mapping mechanisms, please refer to [ID.draft-pentikousis-supa-mapping].

4.3. Network Elements

The Network Element (NE) responds to requests and commands from the NM/NC and makes corresponding configuration changes. An NE may be a physical or a virtual entity, and is locally managed (e.g., via CLI, SNMP, or NETCONF).

SUPA will specify mechanisms, in order to enable the NEs to interact with either local or remote network management systems. The

interaction may include the exchange of information, such as configuration and status information. The NEs will be able to push this information in an event that the NM/NC can subscribe to, and/or provide this information after receiving a request from the NM/NC.

5. Security Considerations

Security is a key aspect of any protocol that allows state installation and extracting of detailed configuration states. More investigation remains to fully define the security requirements, such as authorization and authentication levels.

6. IANA Considerations

No IANA considerations.

7. Acknowledgements

The authors of this draft would like to thank the following persons for the provided valuable feedback: Diego Lopez, Jose Saldana, Spencer Dawkins, Jun Bi, Xing Li, Chongfeng Xie, Benoit Claise, Ian Farrer, Marc Blancet, Zhen Cao, Hosnieh Rafiee, Mehmet Ersue, Mohamed Boucadair, Jean Francois Tremblay, Tom Taylor, Tina Tsou, Georgios Karagiannis, John Strassner, Raghav Rao, Jing Huang.

Early version of this draft can be found here:
https://tools.ietf.org/html/draft-zhou-supa-architecture-00
At the early stage of SUPA, we think quite some issues are left open, it is not so suitable to call this draft as "architecture". We would like to rename it to "framework". Later there may be a dedicated architecture document.

8. Normative References

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

9. Informative References

[I2RS] Interface to the Routing System (i2rs) charter, http://datatracker.ietf.org/wg/i2rs/charter/

[ID.draft-ietf-netconf-restconf] A. Bierman, M. Bjorklund, K. Watsen, R. Fernando, "RESTCONF Protocol", IETF Internet draft (work in progress), draft-ietf-netconf-restconf-04, January 2015

[ID.karagiannis-supa-problem-statement] G. Karagiannis, Q. Sun, L. M. Contreras, P. Yegani, JF Tremblay, " Problem Statement for Simplified Use of Policy Abstractions (SUPA)" IETF Internet Draft (work in progress)", draft-karagiannis-supa-problem-statement-06, March 2015.

[ID.draft-cheng-supa-ddc-use-cases] Y. Cheng, JF. Tremblay, J. Bi , L. M. Contreras, "Use Case for Distributed Data Center in SUPA", IETF Internet draft (Work in progress), draft-cheng-supa-ddc-use-cases-06, April 2015

[ID. draft-zaalouk-supa-vpn-service-management-model] D. Zhang, A. Zaalouk, K. Pentikousis, Y. Cheng, "VPN Service Management YANG Data Model", IETF Internet draft (Work in progress), draft-zaalouk-supa-vpn-service-management-model-03, April 2015

[ID.draft-strassner-supa-generic-policy-info-model] J. Strassner, "Generic Policy Model for Simplified Use of Policy Abstractions (SUPA)", IETF Internet draft (Work in progress), April, 2015

[ID.draft-bi-supa-policy-model] J. Bi, R. Tadepalli, M. Hayashi, "DDC Service Policy YANG Data Model", IETF Internet draft (Work in progress), March, 2015

[ID.draft-pentikousis-supa-mapping] K. Pentikousis, D. Zhang, "Simplified Use of Policy Abstractions (SUPA): Configuration and Policy Mapping", IETF Internet draft (Work in progress), draft-pentikousis-supa-mapping-04, March 2015

[NETCONF] Network Configuration (netconf) charter, http://datatracker.ietf.org/wg/netconf/charter/

[RFC6020] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6991] J. Schoenwaelder, "Common YANG Data Types", RFC 6991, July 2013.

[RFC6241] R. Enns, M. Bjorklund, J. Schoenwaelder, A. Bierman, "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.

Authors' Addresses

Cathy Zhou
Huawei Technologies
Bantian, Longgang District
Shenzhen 518129
P.R. China

Email: cathy.zhou@huawei.com

Luis M. Contreras
Telefonica I+D
Ronda de la Comunicacion, Sur-3 building, 3rd floor
Madrid 28050
Spain
Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://people.tid.es/LuisM.Contreras/

Qiong Sun
China Telecom
No.118 Xizhimennei street, Xicheng District
Beijing 100035
P.R. China

Email: sunqiong@ctbri.com.cn

Parviz Yegani
JUNIPER NETWORKS
1133 Innovation Way
Sunnyvale, CA 94089
Email: pyegani@juniper.net