SUPA
Internet-Draft
Intended status: Informational
Expires: February 6, 2017

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

August 5, 2016

Applicability of SUPA
draft-cheng-supa-applicability-00.txt

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on February 6, 2017.

Table of Contents

1. Introduction

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

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

Three generic models are defined in [I-D.strassner-supa-generic-policy-info-model]: Generic Policy Model, Eca Policy Rule Model, Logic Statement Model. The ECA information model is intended for dynamic service automation; while the Logic Statement Model is intended for expressing high requirements without being involved in network details.

Data models can be defined by developers / operators or by any third party, as long as they follow the common concepts defined in SUPA generic model. [I-D.chen-supa-eca-data-model] defines a policy data model of Event-Condition-Action (ECA), which is an example.

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

2. Terminology

DC Data Center

PCE Path Computation Element

SES Switched Ethernet services

SP Service Provider

SUPA Simplified Use of Policy Abstractions

VM Virtual Machine

VPC Virtual Private Cloud

3. Framework

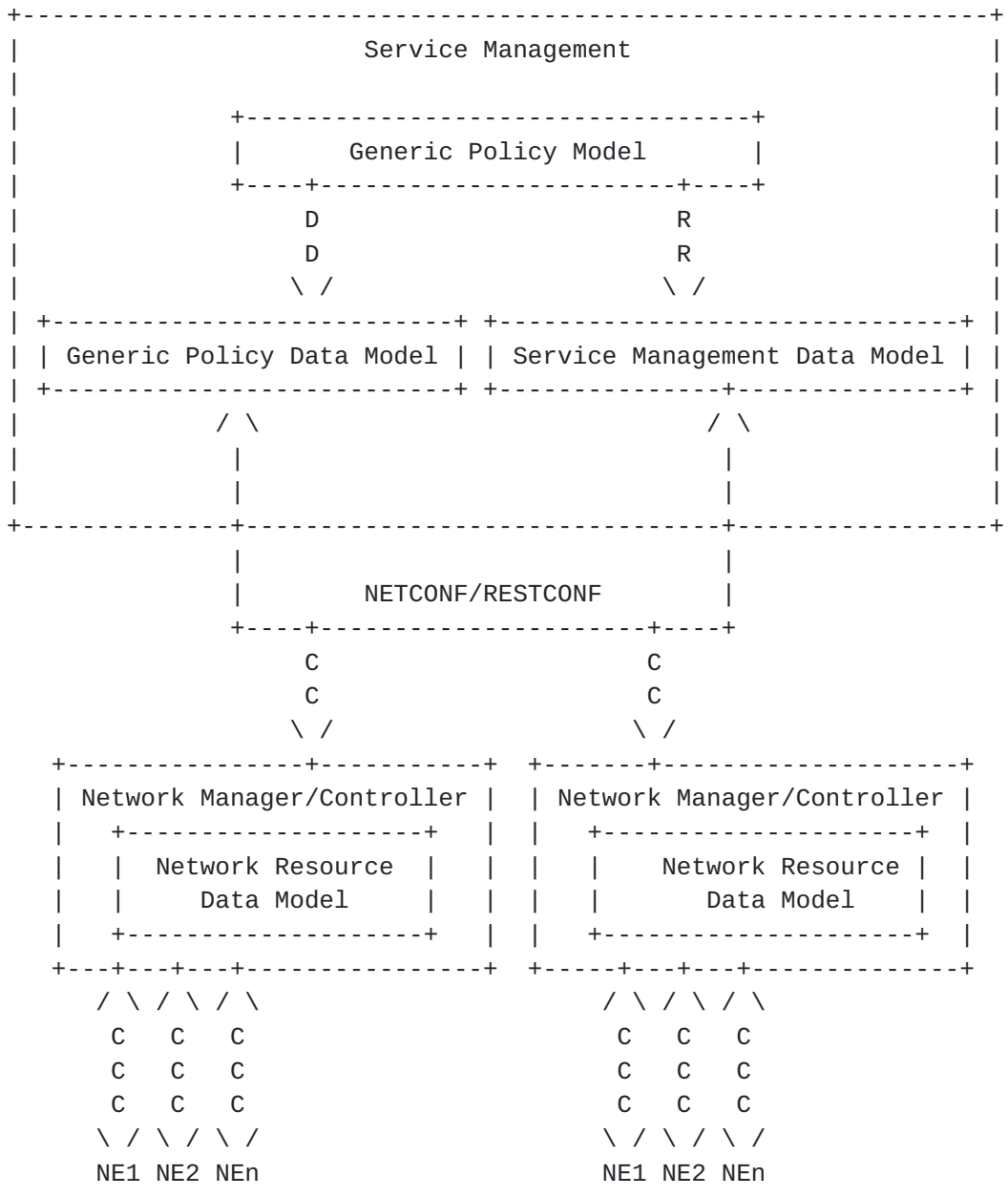

Figure 1: Use of SUPA Models

C: Communications

D: Derived from

R: References (i.e., the generic model is used by the system to instantiate the data model).

As shown in Figure 1, SUPA will define generic policy models, which are independent of services and use cases. Policy data models can be derived from the generic models. The data model will define high level, maybe network-wide policies. Policy data model will be used

in conjunction with service data models to generate configurations for network elements. The service data model is use case specific and will be developed by operators or third parties, which is out the scope of SUPA.

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

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

3.1. Network Manager/Controller

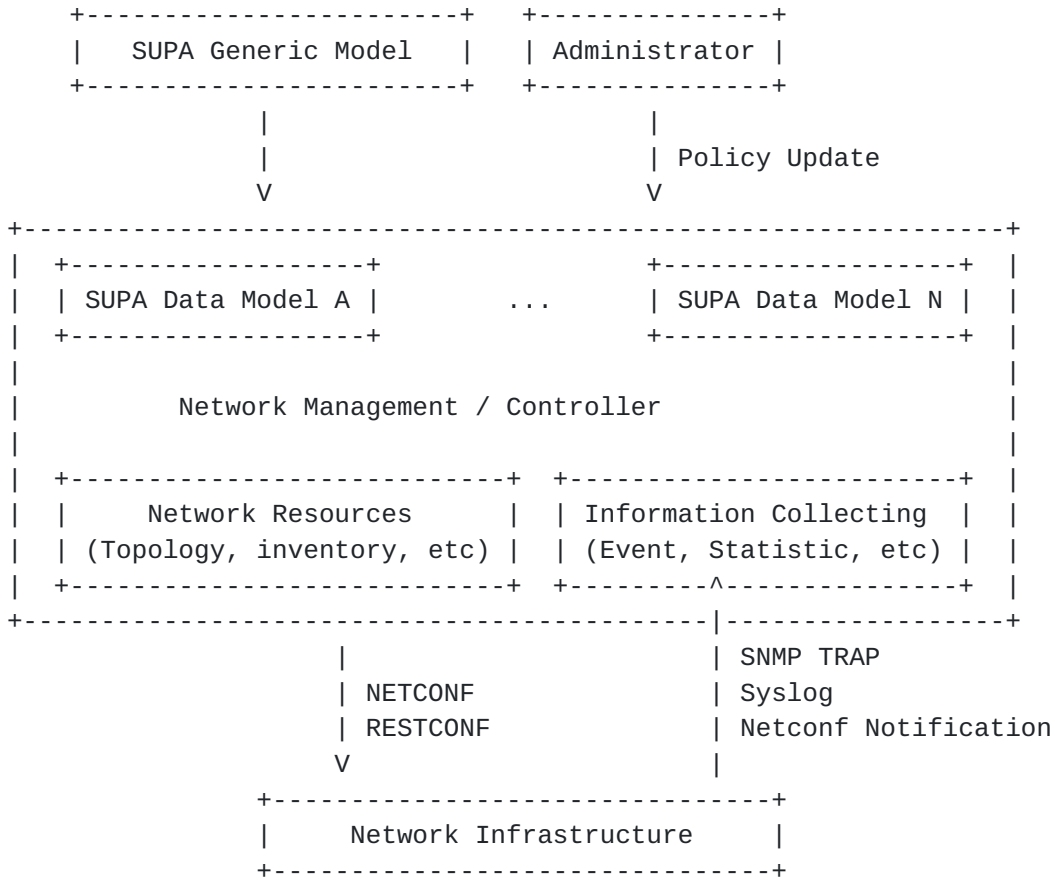

Figure 1: Network Manager / Controller

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

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

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

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

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

For example, there is a configuration request:

All edge routers shall have SSH disabled.

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

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

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

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

Event: link_load > threshold

Condition: there are disjoint links

Action: perform traffic steering

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

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

In the previous example shown below again:

All edge routers shall have SSH disabled.

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

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

4. Use Cases of SUPA

4.1. Use Case 1: SES

4.1.1. Scenario

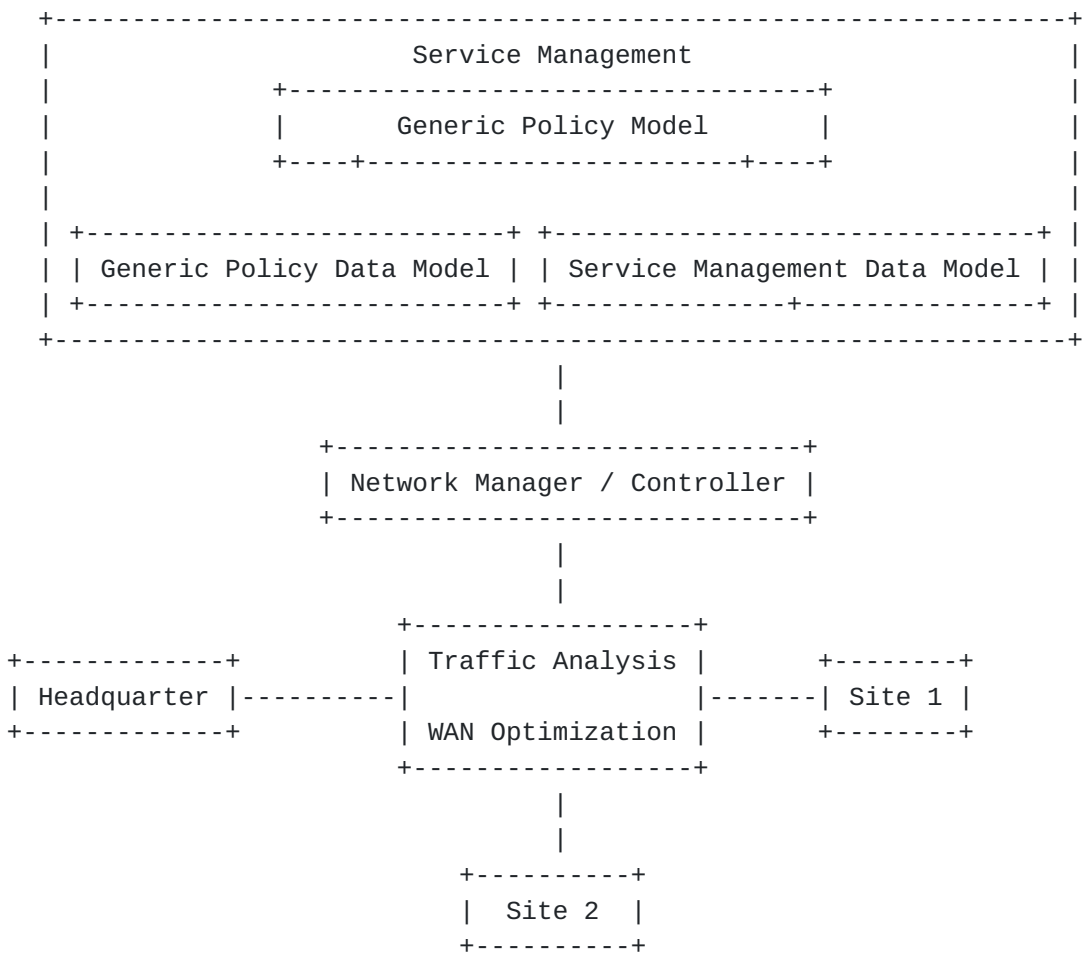

Figure 3: Switched Ethernet Service

Switched Ethernet services (SES) to Small and Medium Businesses business is a growing business segment of the service provider. As the Enterprise's applications grow in demands in terms of the bandwidth and richness of applications, WAN optimization is needed to improve the service quality. SUPA policy data models can be used for maximizing the WAN performance by analyzing the traffic and performing application management and acceleration tools for the network.

In the use case below, 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.

While speed and bandwidth are at the forefront of the WAN Optimization there need to be tools in place to detect, diagnose, remedy and report application performance to ensure the SLAs for a customer are enforced.

The service is modeled in terms of what kind of service (Ethernet, VLAN), bandwidth (10Mbps- 10 Gbps), service package (platinum, gold, silver) etc.

Policy models are based on an Event condition action like:

1. Bandwidth usage alarm triggers data caching

2. Latency alarm triggers reduction of re transmission

3. WAN outage at a specific site can trigger geographic redundancy (provided the service is setup for GR)

The above are 3 of the primitives (Event condition action - ECA) on which the run time operations could be based on. When the service model is comprehensively designed with more possibilities (variables), more policy models could be implemented

4.1.2. Generic Policy Models

Requirements and configurations derived from above application scenarios can be described by service data model and policy data models as below:

Service data model can be used to describe attributes for the SES, including service package type (Platinum, gold etc), bandwidth bought by the subscriber (100Mbps, 10Gbps), connection name -copper/ GigE, latency, etc.

Policy data model describes a condition when the link capacity reaches 90%, Service prioritization and WAN optimization need to be enforced based on the customers service package. Event is the link utilization and condition is the usage and action is the WAN optimization. The actions could trigger multiple actions like data compression, protocol acceleration (like streaming gets priority) which are beyond the scope of SUPA.

ECA Policy:

Event: link_load > 90%

Condition: acceleration for service available

Action: data compression; protocol acceleration

It is assumed that the network management/controller module has the network topology and monitors the load on links in the topology.

When translating and processing the SUPA data model, the link information, including link attributes and load, will be provided by the network management/controller. If the load on a specific link exceeds a threshold, the network manager/controller will trigger actions specified in the model.

The actual actions may be vendor specific, network management/ controller specific or device specific. The actions will be mapped into configuration for network devices. The network management/ controller also need to figure out the set of network devices which need to be configured based on network topology together with some other information, such as service specific information. This is the internal functions of network management/controller, which is out of the scope of SUPA.

4.1.3. Programmatic approach - SUPA modeling

The advantage of the programmatic approach can be maximized by defining as many SUPA ECA models as possible in a top down approach.

In this use case, since this is a switched service, point to point traffic can be identified (by IP Address and port number) and segmented and whole bandwidth can be utilized by many applications simultaneously. Examples are: Print jobs, backups etc..

The benefit of the SUPA is in creating many policies upfront. As the operations grow in complexity SUPA can expand an existing policy by adding more variables. This is how reusable policies can be developed upfront and configuration and maintenance operations can be dealt by modeling and programmatic approach.

Logic Statement Model can also be called as declarative or intent model. This type of model will describe the service intention without specifying low level details, such protocol level or network device level detail, but just the service requirements itself.

4.1.4. SUPA Data Model for SES Use Case

The following model segment is based on [I-D.chen-supa-eca-data-model].

In the model, the event can be expressed using some standardized names, such as the SNMP TRAP (linkDown, linkup, Failure, etc), or "link-load > 90%".

The condition(s) can be expressed using script, such as Python script hasAcceleration("ses") or Python script hasDisjointLinks(DC1, DC2). The script is supposed to be interpreted by a script tool and there

are various script tools, the implementer can use any one as they like, either an existing one like Python or a new one. The script itself is out the scope of SUPA; a simple value will be return by the script tool. Some complex combination of conditions can be expressed using script which will give more flexibility.

When handling the condition script, the script tool will be called to process the script. In this case, the script will communicate with service management system and/or the tenant database to find out if any optimization is available for this service or tenant.

Script can also be used for actions.

An example of the script using Python is:

```
    service-name="ses"

    // input: service-name, type: string

    // output: enhancement, type: string or None if no enhance

    def queryEnhanceinCapability(service-name):

      for i in range(len(capability-models)):

        if getServiceName(capability-models[i]) == service-name:

          return getEnhance(capability-models[i])

    return None


    // input: service-name, type: string

    // output: True/False, type: boolean

    def hasAcceleration(service-name):

      if queryEnhanceinCapability(service-name) == None:

        return False

      else:

        return True
```

The capability data models are supposed to contain the following:

```
<capability-data-model>
  ...
  <services>
    <service>
      <service-name>ses</service-name>
      <service-enhance>compression</service-enhance >
    </service>
    <service>
      ...
    </service>
  <services>
</capability-data-model>
```

The SUPA XML example is shown below:

```
<supa-policy>
  <supa-policy-name>ses-policy</supa-policy-name>
  <supa-policy-priority>0</supa-policy-priority>
  <supa-policy-validity-period>
    <start>00-00-0000</start>
    <end>00-00-0000</end>
  </supa-policy-validity-period>

  <supa-policy-target>
    <profileType>domain</profileType>
    <asDomainName>operatorA-domain1</asDomainName>
    <businessTypeName>ses</businessTypeName>
    <instance>
      <instanceName>
        // detail to be provided by controller
        <flow-filter>
          <src-ip-addr>10.1.1.0/24</src-ip-addr>
          <dst-ip-addr>20.1.1.0/24</dst-ip-addr>
```

```
            </flow-filter>
            <flow-filter>
              ...... // more filters
            </flow-filter>
          </instanceName>
        </instance>
      </supa-policy-target>

      <supa-policy-atomic>
        <supa-ECA-policy-rule>
          <policy-rule-deploy-status>
            ...... // to be provided by controller
          </policy-rule-deploy-status>
          <policy-rule-exec-status>
            ...... // to be provided by controller
          </policy-rule-exec-status>
          <supa-ECA-component>
            <supa-policy-events>
              <has-policy-events>YES</has-policy-events>
            </supa-policy-events>
            <supa-policy-conditions>
              <has-policy-conditions>YES</has-policy-conditions>
              <conjunctive-type>and</conjunctive-type>
            </supa-policy-conditions>
            <supa-policy-actions>
              <action-execution>YES</action-execution>

          </supa-policy-actions>
          </supa-ECA-component>
        </supa-ECA-policy-rule>
      </supa-policy-atomic>

      <supa-policy-statement>
        <event-list>
          <event-name>
            <eventType>entity</eventType>
            // entity or script or boolean
            <entity>"link-load > 90%"</entity>
          </event-name>
        </event-list>

        <condition-list>
          <condition-linkThreshold>
            <conditionType>script</conditionType>
            // entity or script or boolean
            <supa-script>
              <supa-script-content>hasAcceleration(ses)</supa-script-
    content>
```

```
                <supa-script-type>Python</supa-script-type>
                // Python or Perl or any other script
              </supa-script>
            </condition-linkThreshold>
          </condition-list>

          <action-list>
            <actionName>data compression</actionName>
            <actionName>protocol acceleration</actionName>
          </action-list>
        </supa-policy-statement>
      </supa-policy>
```

The data model can be augmented according to developers' need. The developers can add vendor specific events, conditions and actions via "augment" Yang function in [RFC6020], as suggested in [I-D.chen-supa-eca-data-model].

An example of of augmented model is shown below:

```
// ------- yang model snippet start -------
   augment "/supa:supa-policy/supa:supa-policy-statement/supa:event-
   list" {
     leaf my-event{
       description "customized event";
     type bool;

     }
   }

   augment "/supa:supa-policy/supa:supa-policy-
   statement/supa:condition-list" {
     container my-condition{
       description "The bandwidth threshold, unit is Mbps";
     type uint32;
     }
   }

   augment "/supa:supa-policy/supa:supa-policy-statement/supa:action-
   list" {
     container my-action-drop{
     description "drop packets";
     type string;
     }
   }
```

```
      // ------- yang model snippet end -------


      // ------ xml model snippet start -------

            // assume the above augmentation is in a name space "mymodel"
      <supa-policy>
        ......  // others

        <supa-policy-statement>
          <event-list>
            <event-name>
              ......  // other events
            </event-name>
            <mymodel:my-event>
              true
            </mymodel:my-event>   // added event
          </event-list>

          <condition-list>
            <condition-linkThreshold>
              ...... // other conditions
            </condition-linkThreshold>
            <mymodel:my-condition>
              32
            </mymodel:my-condition>   // added condition
          </condition-list>

          <action-list>
            <actionName>

              ...... // other actions
            </actionName>
            <mymodel:my-action-drop>
              drop
            </mymodel:my-action-drop> // added action

          </action-list>
        </supa-policy-statement>
      </supa-policy>

      // ------ xml model snippet end ------
```

4.2. Use Case 2: VPC

4.2.1. Generic

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

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

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

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

tenant may be located in different geographic regions and the VPNs to those VPCs may need to be generated at run time.

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

4.2.2. Example 1

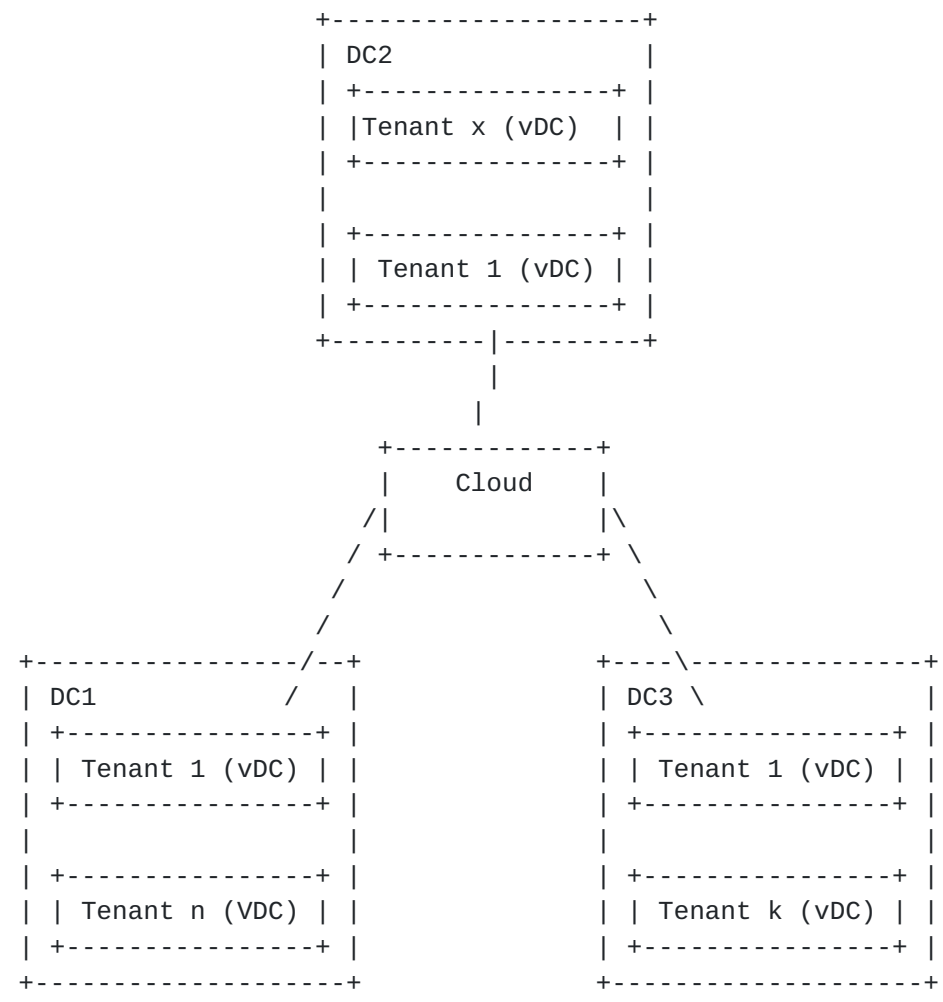

Figure 4: Resource Inter-connection for a VPC Tenant

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

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

The service management system will have a repository of available resources, including the topology. And also the management system will have the customer specific information (location, SLA, agreed resources, etc).

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

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

Declarative policy:

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

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

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

4.2.3. Example 2

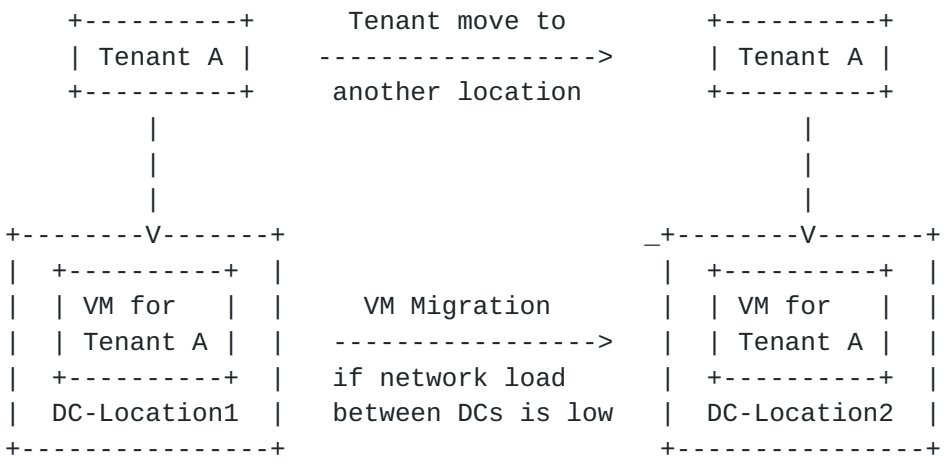

Figure 5: VM Migration if Tenant Move

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

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

ECA Policy:

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

Condition: network_load(DC_old, DC_new) < threshold.

Action:

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

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

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

The user's location can be identified by the user's IP address. When a user login, the network management/controller will check the user's

IP address against an IP address database, such as the IP address assignments by IANA.

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

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

4.3. Use Case 3: Traffic Manipulation cross DCs

DCs usually have multiple external links, either to other DCs or to the internet. Because of the dynamic nature of network traffic, the load on a link may vary at different times of a day, e.g. link mainly carries enterprise traffic may have a high load in the working hours but less traffic in the night. Some events may also impact the load of links, such as one link is physically damaged and the load in it will go to another link.

In order to make full use of the bandwidth of the links, dynamic traffic steering is necessary for SLA meanwhile with full use of network resource.

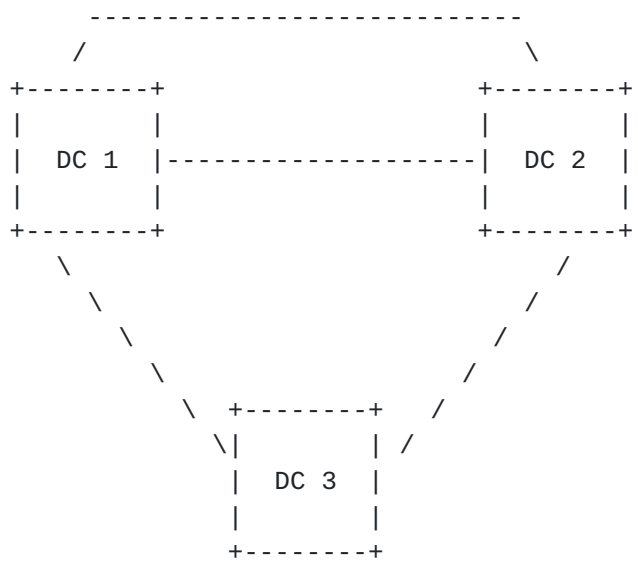

Figure 6: Multiple Disjoint Links Between DCs

Target: a DC has multiple external links. When the load on a link is over a threshold, perform traffic steering for a better bandwidth resource usage

ECA Policy:

Event: load on a DC link exceeds threshold.

Condition: multiple disjoint links between DCs.

Action: steer some traffic to link with low load.

In the above case, it is assumed that the network management/ controller has the network topology, including attributes of the links, such as bandwidth. The network management/controller also monitors the real-time load on the links in the network topology. The network topology also contains the connections between network devices. The network management/controller will be able to figure out if there are multiple disjoint links between two DCs. The algorithm for finding out disjoint links is out of the scope of SUPA.

When the network management/controller detects the load on a link exceeds a threshold, it can check if there are multiple disjoint links, and if yes, it will then further perform necessary actions as pre-specified.

4.4. Use Case 4: Virtual SP

Virtual network operators usually do not build all networks, including access network, metro network, and backbone network, by themselves. Instead, they rent network from other operators. For instance, a virtual operator may not have the access network, traffics of broadband network subscribers will go through an access network rent from another operators, and then be directed to the virtual operators network from the BNG via tunnels. In some another case, a virtual operator may not have the backbone network, the network islands and DCs will be connected by tunnels.

In above cases, virtual network operators may have to face an issue. That is, they have no control over the tunnels and cannot decide the exact path that a tunnel should go through. In some scenarios, if a tunnel goes through the border of two network operators, or the tunnel goes through an area where network load is too high, the SLA may become a problem. Due to cost issue, virtual network operators cannot buy service from other operators with critical SLA. This problem will be even more serious to the a virtual network operator who runs its business in a large geographical region.

A possible solution for such a virtual network operator is to rent or put some routers in network operators' DCs, and then configure tunnels between the routers and perform traffic steering. In this way, virtual network operators can have control over the tunnels, pin down the path. When a problem is detected, such as QoS of a tunnel is below a threshold, virtual network operator can perform "network wide" optimization, reconfigure the tunnels and/or perform traffic steering.

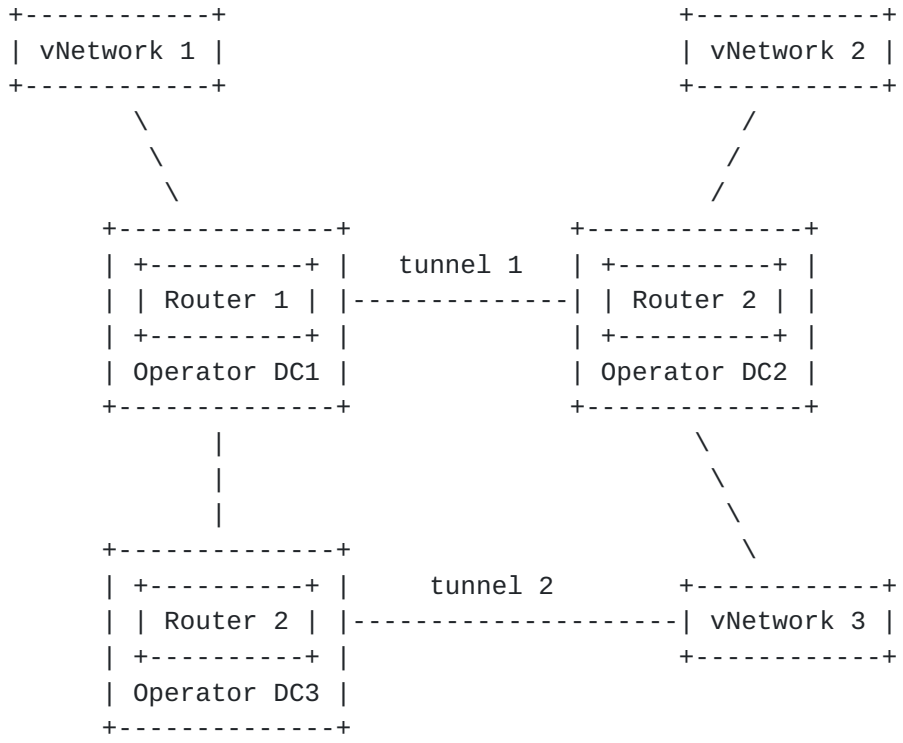

Figure 7: Segment Tunnels for Virtual Network Operator

Assume the route of a direct tunnel built between virtual operator's networks (e.g. vNetwork1-to-vNetwork3) is out of control. For instance, the route may go through network node with problems, or the route may go across the border of different operators where QoS cannot be guaranteed.

In this case, the virtual network operator can configure three tunnels rather than one to connect vNetwork1 to vNetwork3: vNetwork1-to-Router1, Router1-to-Router2, Router2-to-vNetwork3.

After the initial network configuration is finished, if any problem is detected in any tunnel, the network management system can perform network wide optimization, taking all the routers into account and working out another set of tunnels if necessary.

ECA Policy:

Event: QoS parameters < threshold.

Condition: multiple disjoint tunnels available.

Action: Network wide tunnel optimization + traffic steering.

In this case, the virtual SP can monitor the real-time QoS parameters between the virtual networks and the rented routers. If the QoS parameters exceed a threshold, and the virtual has deployed multiple rented routers which can provide multiple disjoint tunnels, then the network management/controller can trigger network wide tunnel optimization and/or perform traffic steering.

When performing the tunnel optimization, the network management/ controller may terminate the tunnel(s) which go through specific network area with problems, and/or build new tunnels, and/or perform network wide traffic steering. This will give the operator a lot of flexibility in controlling the network.

The traffic steering may need to be combined with the network topology, and dynamically distribute traffic in the whole network.

4.5. Use Case 5: Instant VPN

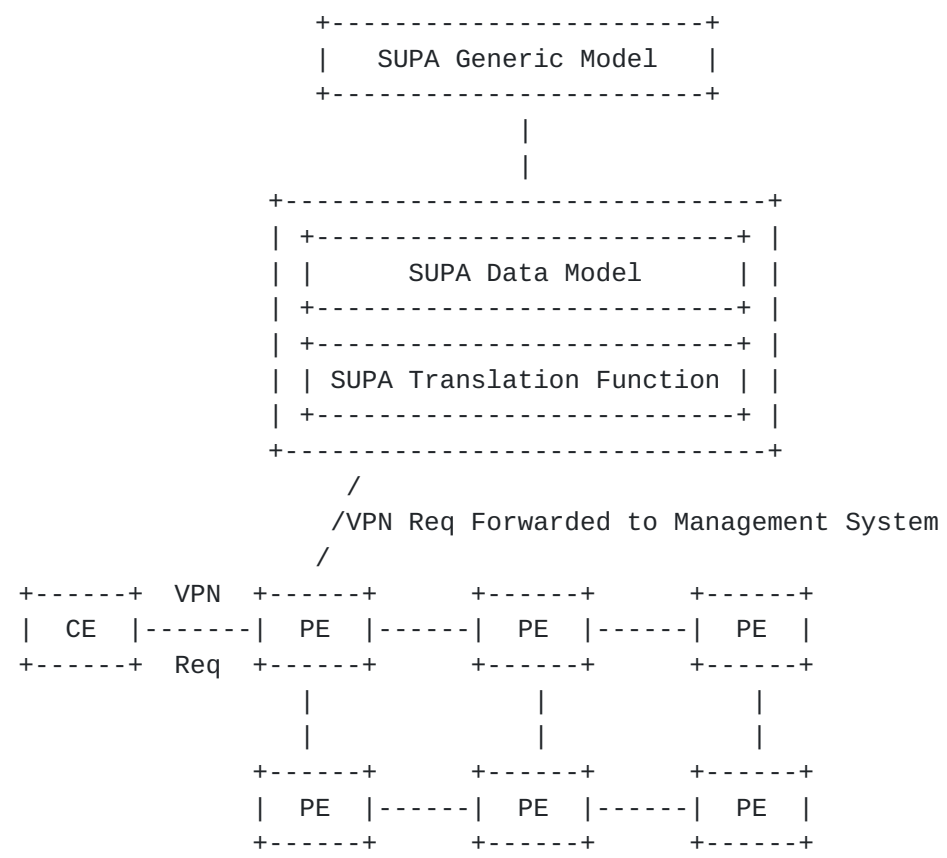

Figure 8: Instant VPN

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

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

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

ECA Policy:

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

Condition: Authentication and Authorization results are OK.

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

4.6. Use Case 6: traffic optimization and Qos assurance on ISP DC

ISPs usually build DCs at the core network border, DCs have more than one uplinks to DC core network; users at MAN can access the core from different links too. Different links may have unbalanced load because of the nature of network traffic. In order to provide service assurance for import tenant, network administrators need to schedule the traffic in specific periods. Traditional network management is usually complex with a long cycle, so it is difficult to meet the request of real-time traffic optimization.

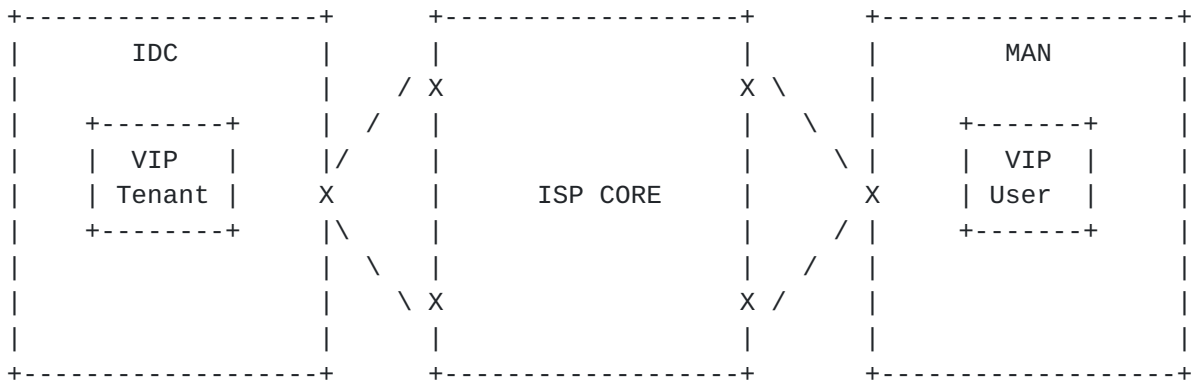

Figure 9: traffic optimization and Qos assurance on ISP DC

Assume that a network controller or orchestrator can monitor network topology, including real-time link utilization and flow information. When utilization of a link reaches a certain threshold, specific flows should be steered to a low load link according to IP address and AS number.

Large service provider's traffic usually has time characteristics, for example, online big sales, network administrator can provide bandwidth assurance for important tenant according to their IP address.

Target 1: a DC has multiple external links. When the load on a link is over a threshold, perform traffic steering for a better bandwidth resource usage.

ECA Policy:

Event: load on a DC link exceeds threshold or a VIP tenant needs bandwidth assurance.

Condition: DC has multiple external links.

Action: steer VIP's traffic to link with low load in a specific period.

Target 2: Tenants or users may have critical request on network Qos. When there are enough bandwidth along the link, perform resource reservation for VIP's traffic on specific links.

ECA Policy:

Event: Tenants or users have critical network requests.

Condition: Resources along the link are enough for reservation.

Action: perform resource reservation for VIP's traffic on specific links.

5. IANA Considerations

This document makes no request of IANA.

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

6. Security Considerations

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

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

7. Acknowledgements

This document has benefited from reviews, suggestions, comments and proposed text provided by the following members, listed in

alphabetical order: Juergen Schoenwaelder, John Strassner, James Huang.

8. Normative References

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

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

Authors' Addresses

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

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

Dapeng Liu
Alibaba Group
Beijing 100022
China

Email: max.ldp@alibaba-inc.com

Borui Fu (Editor)
China Telecom
Beijing
China

Email: fubr@ctbri.com.cn

Dacheng Zhang
Freelancer
Beijing
China

Email: Dacheng.zhang@gmail.com

Narasimha Vadrevu
VN Telecom Consultancy

Email: vadrevun@von20.com