

Network Working Group
Internet-Draft
Expires: April 13, 2012

P. Marques

L. Fang
Cisco Systems
P. Pan
Infinera Corp
A. Shukla
Juniper Networks
October 11, 2011

Traffic classification, filtering and redirection for end-system IP
VPNs.
draft-marques-sdnf-flow-spec-00

Abstract

When IP VPNs are used to interconnect end-systems [I-D.marques-l3vpn-end-system] it may be desirable to introduce traffic control rules at a finer level of granularity than an IP destination address.

This document extends the end-system IP VPN specification with support for fine grain traffic classification, filtering and redirection rules. It applies the existing BGP IP VPN flow specification dissemination mechanism [RFC5575] to end-system IP VPNs in order to provide the ability to control IP packets that match a specific pattern, which may include fields other than the IP destination address.

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 13, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. End-system functionality	6
3. XML schema	8
4. Signaling gateway functionality	10
5. Top-of-rack switch	11
6. Applications	12
7. Security Considerations	13
8. References	14
Authors' Addresses	15

1. Introduction

When end-system IP VPNs [I-D.marques-l3vpn-end-system] are used to interconnect Virtual Machines or other multi-tenant applications it may be desirable to control the flow of traffic between sender(s) and receiver at a finer level of granularity than an IP destination host prefix.

In the IP protocol model, ingress points map traffic into forwarding equivalence classes (FECs) which are then given consistent treatment through a transport network. This document defines a signaling protocol that conveys traffic classification rules. These rules can be applied by ingress points into an end-system IP VPN in order to define FECs that depend on both the destination IP address of the traffic as well as additional fields such as the transport protocol and ports.

One example where this may be desirable is in scenarios where different VPNs may exchange traffic directly. For instance, a VPN that provides a common service to multiple tenants. In this case, the owner of the destination address may wish to inject a traffic rule that limits traffic to TCP packets to and from a specific port. Another example is an application that request specific diffserv [RFC2474] markings for certain types of traffic. In other situations, network administrators may wish to inject specific rules that temporarily redirect traffic.

This document uses a point-to-multipoint model for traffic filtering rules where the traffic egress requests all the ingresses to perform a given traffic classification action. The entity that advertises the destination address of the traffic, or a proxy in its behalf, injects a flow-based route advertisement into the signaling infrastructure. This flow-based route is propagated according to VPN policies to all the ingress points of the VPN, the end-systems which contain VMs allowed to access the destination.

The traffic filtering rules are then applied at all the ingress points of the VPN. The egress MAY also choose to apply the same rules in cases where they are equivalent at both locations.

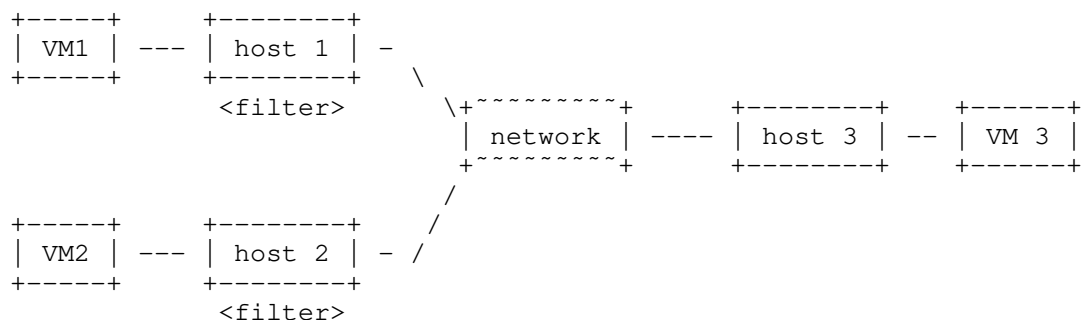


Figure 1

The figure above contains an example topology in which a given VM (VM 3) provides a common infrastructure service. VM1 and VM2 belong to different tenants and are in VPNs which are allowed to access the service in VM3.

This specification allows VM3 to advertise a traffic filtering rule, as a flow-spec route, requesting the Host OSEs in host 1 and host 2 to limit any traffic flow to VM3's destination IP address such that, for instance, only packets for a specific TCP destination port are allowed.

It is important to note that traffic filtering does not avoid the need for application level authorization and authentication.

When a flow-spec route is advertised, the number of possible ingress points is not known in advance. There is no mechanism to generate a positive or negative acknowledgement from the ingress points. This is in contrast to the more traditional network management operation in which the management station is aware of all the agents that must be controlled.

As with the base end-system IP VPN specification, the forwarding and signaling networks are distinct. Flow-spec routes are advertised by the egress end-system or by a proxy in its behalf. The routes are injected into one or more XMPP signaling gateways and propagated using the BGP flow-spec address family [RFC5575].

Using the same vrf-import and export policies that define the IP VPN, the flow-spec routes are then imported from BGP into a vpn-specific database and advertised to all the ingress end-system, which apply them.

This document limits itself to "stateless" traffic classification rules that classify a given IP packet independently of any previous

data traffic.

2. End-system functionality

It is common for end-systems to support traffic classification . One such example is the Linux "ipchains" functionality. This document assumes that such functionality can be associated with a particular Virtual Routing and Forwarding (VRF) table on the end-system and that each virtual interface is associated with a VRF. The traffic classification rules described in this document are applied at the VRF level.

The BGP Flow Specification [RFC5575] document lists a set of IPv4 protocol header fields and match operations that are though to be a minimum common set of supported functionality among hardware implementations.

These fields are:

- o IPv4 destination address.
- o IPv4 source address.
- o IP protocol identifier.
- o Transport Ports: Source, Destination or Either.
- o ICMP Type and Code.
- o TCP flags.
- o Packet length.
- o Diffserv Code Point.
- o IPv4 fragmentation flags.

When numeric values are specified (i.e. fields other than IP addresses), the match operator can specify a list of values with inequality operators. Note that this may result in one logical rule, as defined by this specification to be implemented as multiple classification rules on the underlying OS implementation. For instance the match operations in the Linux "ipchains" implementation are more restrictive.

The match operator is defined via the following BNF grammar:

```
<match> ::= <terms>

<terms> ::= <term>
           | <term> "||" <terms>
           | <term> "&&" <terms>

<term> ::= <operator> value

<operator> ::= "<" | "<=" | "=" | "!=" | ">=" | ">"
```

As an example, a value range is expressed as: ">= begin && <= end".

The result of a flow-spec rule is one of the following actions:

- o allow
- o deny
- o rate-limit
- o redirect
- o copy
- o log
- o set-dscp

The redirect and copy actions have as a target an FEC which should contain an unique UUID [RFC4122] identifier as well as information regarding the SNPA address and label used for forwarding.

The copy action instructs the system to generate a copy of the original packet and forward to the specified FEC. Both copy and log actions have an additional parameter which controls whether all matching packets or a sample is subject to the specified treatment.

The 'set-dscp' action specifies the DSCP value to be assigned to the outer IP header of the packet, when a packet is encapsulated.

3. XML schema

In the end-system IP VPN [I-D.marques-l3vpn-end-system] specification, IP reachability information is encoded as XMPP "item" information belonging to collection nodes where each collection is the IP reachability information for a given VPN. End-systems can publish and receive notifications for these nodes.

This document uses the same approach. It uses a collection with the name of "<vpn-customer-name>/ip4-flow-spec" to publish and receive updates corresponding to IPv4 flow-spec routes. When an end-system published a node into such a collection it must generate a node name that is unique among the nodes that it publishes. It then associates that node with the collection.

XML encoding used by flow-spec items:

```
<item>
  <entry xmlns='http://ietf.org/protocol/bgpvpn/ip4-flow-spec'>
    <ip4-destination>10.0.1/24</ip4-destination>
    <ip4-source>20.0.128/20</ip4-source>
    <ip4-protocol>=6 || =17</ip4-protocol>
    <port>=80</port>
    <destination-port>=80</destination-port>
    <source-port>=80</source-port>
    <icmp-type>=1</icmp-type>
    <icmp-code>=1</icmp-code>
    <tcp-flags>=(syn|rst|ack|fin)</tcp-flags>
    <ip-length>>40</ip-length>
    <dscp>=0</dscp>
    <ip4-fragment>=(df|first|more|last)</ip4-fragment>
    <action>
      <accept/>
      <deny/>
      <rate-limit rate='10pps' />
      <redirect>
        <fec uuid='550e8400-e29b-41d4-a716-446655440000'>
          <snpa af='1'>'infrastructure-ip-address'</snpa>
          <label>1</label>
        </fec>
      </redirect>
      <copy>
        <fec>...</fec>
        <sample/>
      </copy>
      <log/>
      <set-dscp>128</set-dscp>
    </action>
  </entry>
</item>
```

The sequence of XML elements in an item SHOULD follow the "flow specification" NLRI type order as the example above. IP source and destination prefixes are encoded in their standard textual representation of <dotted notation>"/"<prefix-length>. Protocol and Port elements are expressed using the match operator syntax documented above. "<port>" and "<destination-port>" or "<source-port>" SHOULD be mutually exclusive. The icmp type and code fields as well as ip-length and dscp are again encoded using the value match operator. The ">tcp-flags>" element uses either an equality or match operation of the TCP header flags. A binary match is expressed as "m/(syn|rst|ack|fin)/". The "<ip4-fragment>" element may also use a binary match operation.

4. Signaling gateway functionality

As with IP reachability information, signaling gateways create a routing database for each 'vpn-customer-name'. An XMPP client (an end-system) can publish and subscribe to multiple of these databases. Each "virtual interface" on the end-system is associated with a virtual routing table on the gateway.

From a signaling perspective, the gateway functions as a IP VPN PE as described in section 8 of [RFC5575]. As with IP reachability, this document uses the XMPP interface to delegate the forwarding functionality to the end-system, separating it from the signaling node.

In [RFC5575] no route validation procedure is defined for the IP VPN application. For the purposes of the end-system IP VPN application, signaling gateways SHOULD enforce the following rules.

A flow-spec route is valid if its Route Target list is an exact match to the export route target list for the virtual routing table.

A flow-spec route is valid if it contains an IP destination prefixes and there is an exact match between its Route Target list and the Route Target list contained in the IP unicast route that covers that specific destination prefix.

A flow-spec route should be considered unfeasible otherwise and not imported into the specific virtual routing database.

5. Top-of-rack switch

It may be desirable to implement some of the traffic classification functionality on a traditional network element, rather than in the end-system. For instance the end-system may not fully support all the desired functionality.

In this case, a network element can have access to the signaling information using two different methods:

By receiving BGP signaling information directly. A Top-of-rack switch, for example, could infer whether a given end-system is downstream from it by examining the IP infrastructure addresses of the end-systems and extracting information into its forwarding plane whenever an end-point of a VPN is downstream.

By using a "men-in-the-middle" technique in which the XMPP client sessions from end-systems terminate in the TOP-of-rack switches. The switch can then establish an XMPP session to the signaling gateway and proxy the information between the two sessions.

The second approach presents the switch itself with a simplified interface in which it does not need to understand the policies associated with a specific VPN.

6. Applications

This specification provides a mechanism to distribute traffic classification rules to many enforcement points. This may of interest in applications where it is desirable to avoid the standard approach of a centralized enforcement point. Typically in situations where the volume of traffic or the nature of the problem make it more cost effective to do so.

One such application is the enforcement of stateless traffic forwarding rules for infrastructure services. An application level services, such as a storage server may need to support multiple data-center tenants. In this scenario the storage VPN advertises a given address prefix, which contains both the anycast IP address of the load-balancers as the addresses of individual servers. Using VPN import policies, the data-center management solution allows the tenant specific VPNs to see these routes. The tenant VPN addresses must also be reachable on the storage VPN, in this example.

This specification allows the storage service to block out traffic that does not match the specific transport protocols used to provide this service. It also allows confirming traffic to be marked with the appropriate diffserv classification. The network administrator case also use this mechanism for diagnostic purposes.

7. Security Considerations

There are two independent areas that are worth examining when it comes to security. The integrity of the control plane information and the forwarding actions.

This document assumes that all signaling interactions use mutual authentication, where all communication channels are authenticated.

For traffic filtering and redirection this mechanism assumes a "best-effort" model. The ingress points will strive to perform the actions specified by the egress. However there are no strict guarantees that the actions can be applied successfully on an ingress points or that the order of operations is such that no non-conforming traffic is ever presented to the egress.

For traffic filtering rules, the egress point can choose to apply the rules also in order to provide stronger guarantees.

Applications should themselves authenticate its communication peers my methods that do not depend on the IP addresses used at the network layer.

8. References

- [I-D.marques-l3vpn-end-system]
Marques, P., Fang, L., and P. Pan, "End-system support for BGP-signaled IP/VPNs.", draft-marques-l3vpn-end-system-01 (work in progress), October 2011.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, August 2009.

Authors' Addresses

Pedro Marques

Email: pedro.r.marques@gmail.com

Luyuan Fang
Cisco Systems
111 Wood Avenue South
Iselin, NJ 08830

Email: lufang@cisco.com

Ping Pan
Infinera Corp
140 Caspian Ct.
Sunnyvale, CA 94089

Email: ppan@infinera.com

Amit Shukla
Juniper Networks
1194 N. Mathilda Av.
Sunnyvale, CA 94089

Email: amit@juniper.net

sdnp discussion group
Internet Draft
Intended Status: Informational
Expires: April 21, 2012

D. McDysan
Verizon

October 24, 2011

Cloud Bursting Use Case

draft-mcdysan-sdnp-cloudbursting-usecase-00.txt

Status of this Memo

This Internet-Draft is submitted to IETF 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/lid-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 April 17, 2011.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This draft describes a use case for the overall coordination, control and management of "cloud bursting" in a hybrid cloud computing environment involving a private data center and a public or virtual private multi-tenant data center. This use case may be relevant to the Software Driven Network Protocol [SDN_UC], VPN for Data Center [VPN4DC], or Cross Stratum Optimization [CSO] discussions in the IETF.

Table of Contents

1. Introduction.....	2
2. Conventions used in this document.....	2
2.1. Acronyms.....	2
2.2. Terminology.....	3
3. Motivation and Background.....	3
4. Proposed Use Case.....	3
4.1. General Dynamic Cloud Computing Functionality.....	3
4.2. Private Data Center Use Case Elements.....	5
4.3. Public of Virtual Private Data Center Elements.....	6
5. Security Considerations.....	7
6. IANA Considerations.....	7
7. References.....	7
7.1. Normative References.....	7
7.2. Informative References.....	7
8. Acknowledgments.....	8

1. Introduction

This draft describes the cloud bursting use case for implementing dynamic cloud computing in a multi-tenant environment that addresses the case where computing, storage, application, security, networking resources are dynamically assigned.

Section 3 provides some motivation and background for the cloud bursting use case. Section 4 provides more details on the cloud bursting use case.

The draft cites a number of references that provide further detail on specific aspects of the use case and/or requirements.

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 RFC-2119 [RFC2119].

2.1. Acronyms

SDNP	Software Driven Network Protocol
VPN4DC	VPN for Data Center
CSO	Cross Stratum Optimization

2.2. Terminology

Private Cloud - operated by an enterprise

Public Cloud - multitenant data center operated by a service provider accessed via the "Public" Internet

Virtual Private Cloud - multitenant data center operated by a service provider accessed via a L2 and/or L3 Virtual Private Network (VPN)

Hybrid Cloud - Dynamically instantiated instance of a public or virtual private cloud to (temporarily) augment capacity of a private cloud

3. Motivation and Background

Currently, mostly static L1/L2/L3 networks interconnect customer applications in Private Enterprise data centers. Note that an Enterprise application network may also contain sites that support sensor data collection and other forms of input or output. In order to provide capacity in support of dynamic application demand from Enterprises, cloud service providers are deploying virtualized resources (e.g., processing, storage, apps/OS) in Cloud Computing Centers.

Some dynamic bandwidth on demand being done in access networks, but this is often not automatically coordinated with networking in a cloud data center. Furthermore, assignment, reservation and configuration of other resources needed by the application, such as computing, storage, access to databases, or specific software instances is not well coordinated with the assignment of network capacity. An opportunity exists to standardize methods to optimize the assignment of L1, L2, L3 network capacity, Cloud Computing site selection and/or resource allocation more dynamically.

Such an optimization may be done proactively on a reservation basis, reactively in response to ad hoc requests, reactively in response to detected changes in load using pre-defined policies, or reactively by the provider in response to an aggregate of a number of smaller requests and/or reservations in order to make the system more efficient, and/or prepare for honoring a future reservation.

4. Proposed Use Case

This draft describes a use case for the overall coordination, control and management of "cloud bursting" in a hybrid cloud computing environment involving a private data center and a public or virtual private multi-tenant data center. This use case describes more details for a similar use case described in section 6.2 of [SDN_UC], section 3 of [CSO] or [VPN4DC].

4.1. General Dynamic Cloud Computing Functionality

A cloud computing instance requires control and management at least the following. Further details on use cases and requirements are listed as references for many of the items below.

- o .Layer 2/Layer 3 bandwidth configuration and monitoring (scheduler weight setting, policer setting, reserving bandwidth (e.g., MS-PW), counter collection)
- o .VPN membership (e.g., VLAN, PBB, L2VPN/L3VPN), reachability and any restrictions on communication within the VPN [VPN4DC], [VROM]
- o .Compute resource allocation: virtual machines, virtual memory, OS, software assignment and activation on a physical computer [VROM]
- o .Storage resources: Partition(s) (e.g., Logical Unit Name (LUN)) assigned to physical storage [VROM]

There may also be a need to configure the following to align with the above

- o Firewall rules (e.g., ACLs) and other settings [FWLBDC]
- o Load balancers and settings [FWLBDC]
- o Security functions (encryption) [VDC_SEC], [DVNSEC]
- o Network Address Translation (NAT) settings [DVN_SEC]
- o Dynamic IP and/or L2 address mobility support

Furthermore, the request may also have performance related parameters, such as [CSO]

- o Packet transfer performance between sites (e.g., latency, delay variation, loss)
- o Availability and failure recovery time objectives for classes of resources (e.g., percentage up time and time to recover from an interface failure)
- o Diversity or fate sharing avoidance constraints (e.g., sets of cloud resources are placed in sites that do not share fate for failure events)

A Private Cloud Enterprise customer desires a single unified method to invoke all of the above aspects of a hybrid cloud service in a transaction such that either all aspects are instantiated, or if any aspect cannot be instantiated then the overall transaction will either fail or result in the next step in a capability/performance negotiation. Previously, this unified methods has been called a "Northbound API," and more recently an "Application-to-Orchestrator protocol" [SDN_PS]. In some cases, a negotiation could occur where the SDN system responds with a capability and/or performance indication that is "closest" to what was requested as a next step in a negotiation process. The Enterprise customer could then have the option to accept this offer, or make another request with changed parameters.

A higher-level system could use interfaces (many of them already standardized by the IETF or other SDOs) to implement the above

control/management interfaces to accomplish this objective as illustrated in Figure 1. The SDN controller could be viewed as implementing a set of "plug-ins" [SDN_PS] for controlling the management, control and/or data plane of the various devices listed above (a subset being illustrated in Figure 1).

Alternatively, signaling between some of these elements for implementing some functions and state/configuration communication could be employed, for example, as described in [VPN4DC].

Furthermore, not all of these interfaces need be standardized in all cases. For example, the private cloud site(s) could have its own controller and the cloud provider another controller. The required interaction is then between these data center controllers and the interfaces within the private cloud need not be standardized.

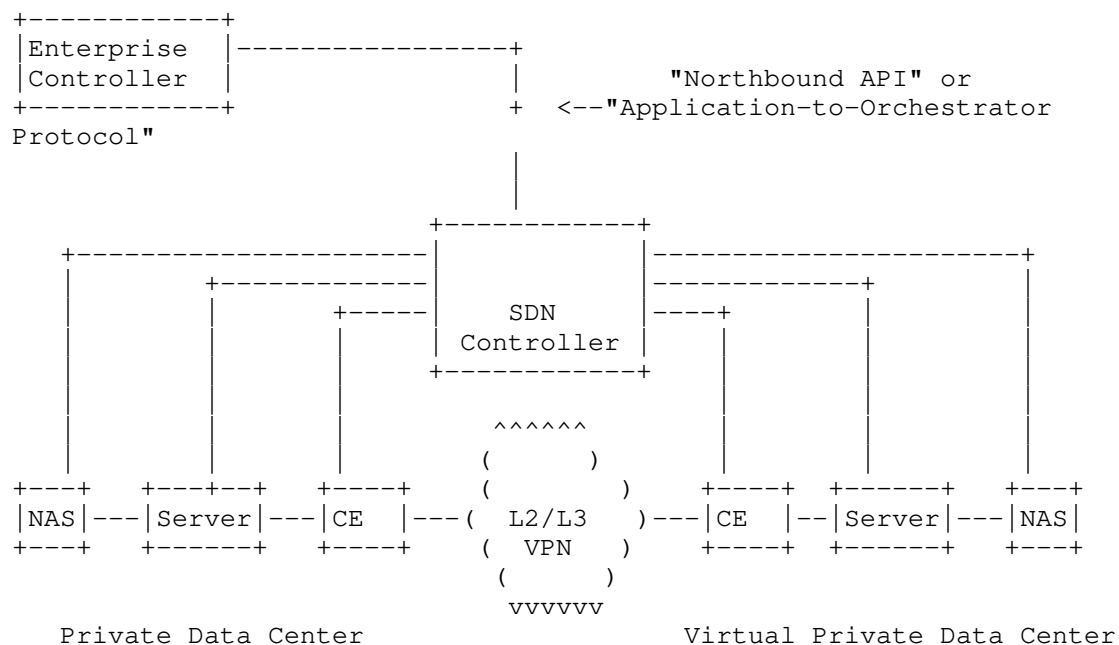


Figure 1 Hybrid Cloud Bursting Use Case Context

4.2. Private Data Center Use Case Elements

Private Data Center controller (may be automated, or a combination of manual and automatic actions) requests the following at one or more private Cloud sites:

- o Configure VM assignment/movement within private cloud
- o Configure storage, LUN assignment/movement within private cloud
- o Configure private cloud switch/router access to networking (e.g., scheduler weights, policer, enable specific L2/L3 addresses)

- o Configuration of any VPN reachability and the requested components from the cloud service provider within the private cloud sites
- o Private cloud Load Balancer, NAT settings
- o Private cloud Firewall, Security function settings
- o Configuration needed to meet performance objectives and/or constraints in Private Cloud Elements.

Usually, the Enterprise operator of the private data center would do all of the above. However, a service provider could do settings for some components. For example, setting the scheduler weights on the switch/router that interfaces to the L2/L3 VPN or Internet access could be done via the service provider.

4.3. Public of Virtual Private Data Center Elements

The SDN controller function of Figure 1 accepts a "Northbound API" request from an Enterprise controller and performs following actions at one or more cloud provider Virtual Private or Public cloud sites.

- o .Configure VM assignment/movement within the Enterprise and provider data center(s). Note that this may require usage of the same or compatible hypervisors with appropriate communication and/or permissions between the hypervisor controllers.
- o .Configure storage, LUN assignment/movement within the provider data center(s). Note that this may require usage of the same or compatible network attached storage systems with appropriate communication and/or permissions between the storage controllers.
- o .Configure bandwidth related characteristics of L2/L3 packet network (e.g., bandwidth for an MS-PW, additional (logical) ports, scheduler weights, policers, addresses). This includes logical connectivity between and enterprise and a provide cloud site as well as between enterprise sites, and between provider cloud sites.
- o .Configure VPN-related characteristics within public or virtual private data center (e.g., mapping to L2/L3 VPN service, firewall)
 - o This may include further definitions of reachability within the L2/L3 VPN or the notion of a Virtual Data Center. See [VPN4DC].
- o .Configure access to networking (e.g., scheduler weights, policer, enable specific L2/L3 addresses) on the Public Virtual Private data center switches or routers
- o Virtual, multi-tenant Private Firewall and security function settings
- o Virtual, multi-tenant Private Load balancer and NAT settings

- o Configuration needed to meet performance objectives and/or constraints. In some cases, the service provider may need to propose an alternative to progress a negotiation if not all objectives or constraints can simultaneously be met. Furthermore, the SDN controller must perform composition across all Enterprise private cloud sites and candidate public or virtual private cloud sites to ensure that the requested performance objectives are delivered.

Usually, the service provider of the public or virtual private cloud data center would do all of the above.

5. Security Considerations

A number of virtual data center security requirements and gaps are described in [VDC_SEC] and [DVN_SEC]. This draft addresses some of these requirements as follows.

Consistent, automatic configuration of VPN membership in the private and public/virtual private cloud is necessary to provide isolation between customers.

Consistent, automatic configuration of firewalls in the private and public/virtual private cloud is necessary to provide fine-grained access control for various virtual data center resources.

Consistent, automatic configuration of security functions (e.g., encryption, authentication, intrusion detection, etc.) in the private and public/virtual private cloud is necessary to provide confidentiality, non-repudiation, and authentication for various virtual data center resources.

6. IANA Considerations

None

7. References

7.1. Normative References

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

7.2. Informative References

[SDN_UC] Ping Pan, Tom Nadeau, "Software-Defined Network (SDN) Problem Statement and Use Cases for Data Center Applications," Work in Progress

[SDN_PS] Tom Nadeau, "Software Driven Networks Problem Statement," Work in Progress.

[VPN4DC] Ning So et al, "Requirements of Layer 3 Virtual Private Network for Data Centers," work in progress.

[CS0] Greg Bernstein, Young Lee, "Cross Stratum Optimization Use-cases," work in progress.

[CLO] Young Lee et al, "Problem Statement for Cross-Layer Optimization," work in progress.

[VROM] V. Grado, T. Tsou, N. So, "Virtual Resource Operations & Management in the Data Center," work in progress

[FWLBDC] Y. Gu, Y.Fan, "Policies and dynamic information migration in DCs," work in progress

[VDC_SEC] S. Karavettil et al, "Security Framework for Virtualized Data Center Services," work in progress

[DVN_SEC] M. Ko, E. Wang, "Problem Statement for Setting Up Dynamic Virtual Network," work in progress

8. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Copyright (c) 2011 IETF Trust and the persons identified as authors of the code. All rights reserved.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

This code was derived from IETF RFC [insert RFC number]. Please reproduce this note if possible.

Authors' Addresses

Dave McDysan
Verizon
22001 Loudoun County PKWY
Ashburn, VA 20147
Email: dave.mcdysan@verizon.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 3, 2012

T. Nadeau, Ed.
CA Technologies, Inc.

P. Pan, Ed.
Infinera, Inc.
October 31, 2011

Framework for Software Defined Networks
draft-nadeau-sdn-framework-01

Abstract

This document presents a framework for Software Defined Networks (SDN). The purpose of the framework is to provide an overall picture of the problem space of SDN and to describe the relationships among the various components necessary to manipulate the components that comprise SDNs. SDN requires the specification of several key components including an "orchestrator" and various "plug-ins" between the orchestrator function and network resources. In addition to this, an orchestrator will require interconnection with standard policy bases, as well as other orchestrators in distributed environments or insofar as to support high-availability and fault-tolerance capabilities. To this end, several interfaces and mechanisms to address issues such as enabling the orchestrator to manipulate and interact with the control planes of devices such as routers and switches, as well as a discourse between different orchestrators will be described. The intent of this document is to outline what each interface needs to accomplish, and to describe how these interfaces and mechanisms fit together, while leaving their detailed specification to other documents.

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].

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 30, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	4
1.1. Terminology	4
1.2. Reference Model	4
2. Building Blocks	6
2.1. SDN Orchestrator	6
2.1. SDN Plug-In	6
3. Overview of SDN Operation	7
4. Main Interfaces	32
4.1. SDN Orchestrator to Application Interface.	32
4.2. SDN Orchestrator Policy Base Interface	33
4.3. SDN Orchestrator to Plug-In Interface.	34
4.4. SDN Orchestrator to Orchestrator Interface	36
4.5. SDN Orchestrator Logging interface	36
4.6. SDN Control Interface	36
5. Deployment Models	37
6. Trust Model	43
7. IANA Considerations	44
8. Security Considerations	44
9. Contributors	45
10. Acknowledgements	46
11. References	46
11.1. Normative References	46
11.2. Informative References	46
Authors' Addresses	47

1. Introduction

The Software Driven Network (SDN) is motivated by several use cases, such as those described in <insert use case draft reference here>. The overall problem space for SDN is described in [I-D.nadeau-sdn-problem-statement] with requirements for solutions found in [I-D.pan-sdn-requirements]. The purpose of this document is to provide an overview of the various components necessary to create SDNs. SDNs require the specification of several interfaces and mechanisms to address issues such as an orchestration point (logical or physical) and interfaces between itself, applications that wish to consume or manipulate network resources, network resources such as router control planes, and policy and security engines. Furthermore, high availability and resiliency mechanisms also need to be defined. The intent of this document is to describe how these interfaces and mechanisms fit together, leaving their detailed specification to other documents.

1.1. Terminology

This document draws freely on the terminology defined in [I-D.nadeau-sdn-problem-statement].

We also introduce the following terms:

SDN Orchestrator: The entity which is the controller of control planes.

Policy Engine or Database: The repository of policy information stored within a network domain.

Location Services: A network service used for locating network elements. Examples include ALTO and The Domain Name System (DNS).

SDN Domain: a host name (FQDN) at the beginning of a URL, representing a set of content that is served by a given CDN. For example, in the URL

http://sdn.example.(com|org|net)/...rest of url...

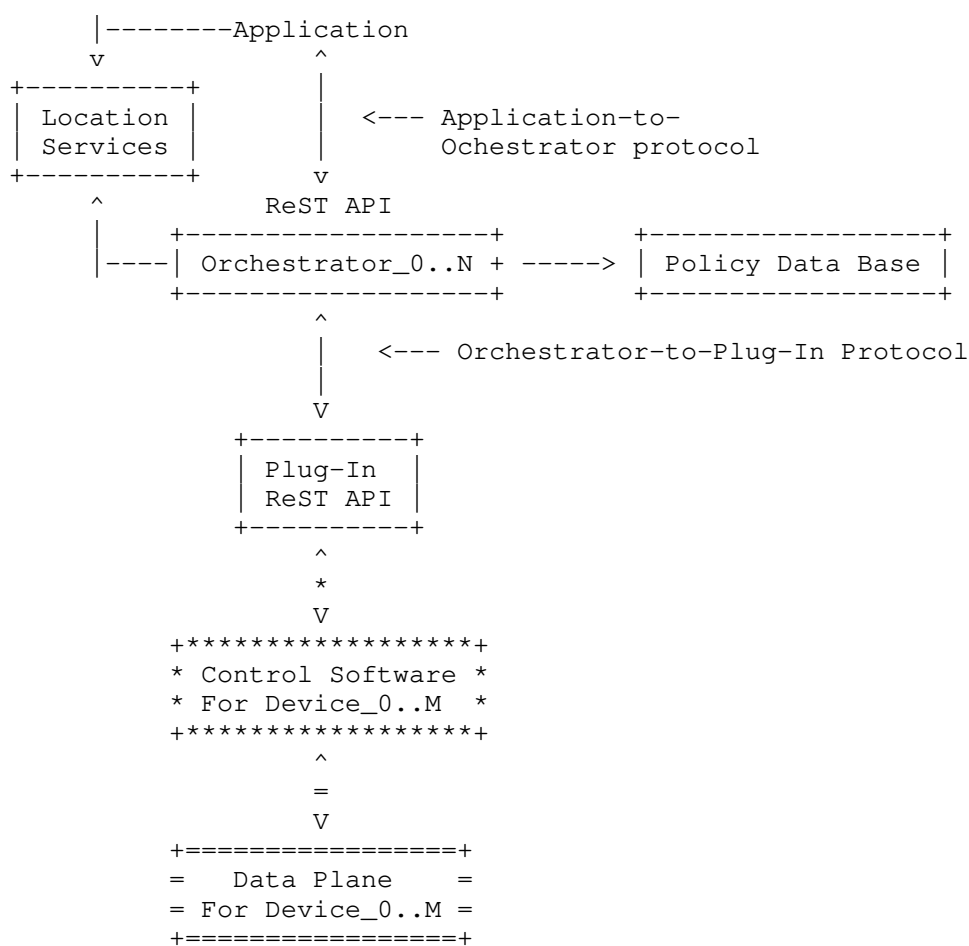
the SDN domain is sdn.example.(com|org|net).

Application Programming Interface (API): is a particular set of rules and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers.

SDN Plug-In: An API that abstracts a device and its object model that an SDN Orchestrator can use to communicate with that device.

1.2. Reference Model

Figure 1 (reproduced from [I-D.nadeau-sdn-problem-statement]) illustrates the basic model of operation with which this document is concerned.



<--> interfaces and objects inside the scope of SDN

+--+

<*> interfaces and objects may be within the scope of SDN

+++ insofar as modifications are needed to support SDN

Figure 1: Model of Operation for SDN

Note that while some interfaces are considered out of scope for SDN, and these are noted above. The overview of operation described below will show how those interfaces are used as part of an overall solution and where new protocols will be defined as well.

2. Building Blocks

2.1. SDN Orchestrator

The controller of control planes, known as the SDN Orchestrator, must be capable of requesting object models from each of the controlling software it is responsible for managing, and therefore each controlling software element must be capable of producing a self-describing object model with which the Orchestrator can use when issuing instructions to manipulate it. Furthermore, communications between the Orchestrator and the control planes must also be rationalized. To this end, the working group will define SDN "plug-ins" which are the abstracted interface to each type of control plane. The working group will also define a protocol that is used for communications between the Orchestrator and the plug-in.

It should be noted that the SDN Orchestrator function is conceptually centralized in that applications SHOULD have a centralized means of locating the Orchestrator within its network. However, in reality the Orchestrator will be designed and architected so as to exist in a distributed manner if so desired operationally. Furthermore, resiliency of a SDN infrastructure and network of components is required as these elements and functional controls are to be used in highly available production environments; therefore, the working group will define mechanisms by which the SDN Orchestrator will operate in this manner as a minimum requirement.

2.2. SDN Plug-In

The SDN Plug-In as shown in the figure above, is an abstraction between the SDN Orchestrator and the network resource or device control plane or "controlling software" to which it is interfacing. The purpose of this interface is as a means of abstracting the controlling software from the device itself. The plug-in is also a means by which the device can negotiate its capabilities with the controller as well as exchange revision information (i.e.: SDN protocol revision identifiers).

3. Overview of SDN Operation

To provide a big-picture overview of the various components of SDN, let us walk through how a typical SDN Orchestrator would be deployed within a network, and then how applications would use it to interact with network resources.

Include 1 use case here... (TBD)

4. Main Interfaces

This section describes the main interfaces between different components of SDN.

4.1 SDN Orchestrator to Application Interface

This interface allows the SDN Orchestrator or "controller" system to be interconnected with applications. This interface is sometimes referred to as a "north-bound" interface, because it points "northward" in architectural diagrams. This interface allow bootstrapping of the interface between the Orchestrator and interested applications. It will allow the applications to authenticate using one or more methods. This interface will allow applications to learn of which objects they have authorization to manipulate, or to interact with objects belonging to controlling software.

4.2 SDN Orchestrator Policy Base Interface

This interface allows the SDN Orchestrator to interconnect with policy, authentication and authorization databases.

4.3 SDN Orchestrator to Plug-In Interface

This interface allows the SDN Orchestrator to interconnect with the controlling software of devices.

4.4 SDN Orchestrator to Orchestrator Interface

This interface allows the SDN Orchestrator to interconnect and interact with other SDN Orchestrators that exist within its SDN Domain.

4.5 SDN Orchestrator Logging interface

This interface allows the Logging system in interconnected SDN

Orchestrators to communicate the relevant activity logs in order to allow log consuming applications to operate in multi-SDN Orchestrator environments. For example, this interface can be used to collect logs from SDN Orchestrators to provide reporting and monitoring to the M/CSP of SDN activities.

SDN Orchestrator logs are easily exchanged off-line as a flat text file, for example, and could include the following information.

- o SDN Domain - the full domain name of the origin server
- o IP address - the IPv4 (and IPv6 if available) address of the client application or SDN Orchestrator making the request
- o Transaction Time - the ending time of the transfer
- o Time zone - any time zone modifier for the end time

4.6 SDN Control Interface

The protocol used between the Orchestrator and another entity such as an application, Policy Database, Location Services or Plug-In, or another Orchestrator in order to send commands, receive replies or emit notifications is the role of the SDN Control Interface.

As noted above and in [I-D.nadeau-sdn-problem-statement], the control interface may also be used for the bootstrapping of other interfaces such as the SDN Orchestrator to SDN Orchestrator interface.

5. Deployment Models

Describe deployment models here. This should include a single domain of Orchestrators and applications, and other components. (TBD)

6. Trust Model

There are a number of trust issues that need to be addressed by a SDN solution. In a standard SDN environment with a single Orchestrator, one policy database, one location services engine (i.e.: DNS/ALTO) and one router associated with the Orchestrator, there are a number of points of trust to consider. First, any of the interfaces between the SDN elements expose

trust points. Further, since the SDN Orchestrator-to-Orchestrator allows for an Orchestrator to bootstrap itself from another's active configuration, the operator must ensure that authorization is configured correctly.

We expect that the detailed designs for the specific interfaces for SDN will need to take these trust issues into account.

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

(Note: this section to be extended in future revision.)

9. Contributors

The following individuals contributed to this document:

o

10. Acknowledgements

We thank ... for helpful comments on the draft.

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.nadeau-sdn-problem-statement]
Nadeau, T., and P. Pan, "
Software Defined Network (SDN) Problem
Statement", draft-nadeau-sdn-problem-statement-00 (work
in progress), September 2011.

[I-D.pan-sdn-requirements]
Pan, P., and T. Nadeau,

"Software Defined Networks (SDN) Requirements",
draft-pan-sdn-requirements-00 (work
in progress), September 2011.

Authors' Addresses

Thomas D. Nadeau
CA Technologies, Inc.
273 Corporate Drive
Portsmouth, NH 03801

Email: thomas.nadeau@ca.com

Ping Pan
Infinera Corporation
169 Java Drive
Sunnyvale, CA 94089
Phone: (408) 572-5200

Email: ping@pingpan.org

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 31, 2012

T. Nadeau, Ed.
CA Technologies, Inc.

P. Pan, Ed.
Infinera

October 31, 2011

Software Driven Networks Problem Statement
draft-nadeau-sdn-problem-statement-01

Abstract

Software Driven Networks (SDN) is an approach to networks that enables applications to converse with and manipulate the control software of network devices and resources. SDNs are comprised of applications, control software, and interfaces to services that are hosted in an overlay or logical/virtual network as well as those possibly same components that comprise the underlying physical network. Modern applications require the ability to easily interact and manipulate these resources. Applications can benefit from knowing the available resources and from requesting that the network makes the resources available in specific ways. To this end, there is a requirement to couple applications more closely to the underlying resources on which they depend, consume and interact with.

SDN infrastructure and components exist in most deployed networks today. Some of these components are being standardized by various organizations, as well as some being already standardized by the IETF. However, no standards or open specifications currently exist to facilitate end-to-end operation of a software defined network, specifically one that provides open APIs for applications to control the network services and functions offered by device control planes or other "controlling" software. The goal of this document is to outline the problem area of SDN for the IETF.

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].

Status of this Memo

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

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

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

This Internet-Draft will expire on September 30, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	5
1.1. Terminology	5
1.2. SDN Background	9
2. SDN Interconnect Use Cases	9
3. SDN Interconnect Model & Problem Area for IETF	11
3.1. Candidate SDN Problem Area for IETF	13
4. Design Approach for Realizing the SDN APIs	15
4.1. Relationship to the OSI network model	15
4.2. "Reuse Instead of Reinvent" Principle	16
4.3. Application to SDN Orchestrator Interface	16
4.4. SDN Orchestrator to Plug-In Interface	18
4.5. SDN Logging Interface	19
4.6. SDN Orchestrator to Location Services Interface	20
4.7. SDN Orchestrator to Policy Database Interface	20
4.8. SDN Orchestrator to SDN Orchestrator Interface.	20
5. Gap Analysis of relevant Standardization and Research Activities	20

5.1. Open Network Forum	21
6. Relationship to relevant IETF Working Groups	23
6.1. ALTO	23
7. IANA Considerations	25
8. Security Considerations	25
9. Acknowledgements	26
10. References	26
10.1. Normative References	26
10.2. Informative References	27
Appendix A. Additional Material	29
A.1. Non-Goals for IETF	29
A.2. Prioritizing the SDN Work	30
A.3. Related standardization activities	31
A.4. Related Research Projects	35
A.4.1. IRTF Cross Stratum Optimizaiton Research Group	35
Authors' Addresses	36

1. Introduction

Software Driven Networks (SDN) is an approach to networks that enables applications to converse with and manipulate the control software of network devices and resources. Modern applications require the ability to easily interact and manipulate resources provisioned and controlled by networks. Applications can benefit from knowing the available resources and from requesting that the network makes the resources available in specific ways.

To this end, there is a requirement to couple applications more closely to the underlying resources on which they depend, consume and interact with. In particular, modern applications require interaction with and the manipulation of both physical and virtual compute, storage and connectivity resources and abstract interfaces to these things. These abstractions must also allow applications to manipulate resources at varying levels of granularity, policy and security. It is also worth noting that modern Software Driven Networks (SDNs) are comprised of applications, control software, and interfaces to services that are hosted in an overlay or logical/virtual network as well as those possibly same components that comprise the underlying physical network.

These services include path computation, topology discovery, firewall services, domain name services, network address translation services, virtual private networks and the like. These services and elements may be physical or virtual.

One requirement is to create a means by which applications can

communicate with the control planes of the underlying network devices or entities which control and own network resources on which they depend. Note that the "control planes" of network devices will be referred to as "controlling software" to abstract away the concept of the software that controls a device's data plane. The control planes of devices, whether physically co-located with a device and its data plane, or externally located for example in the case of an OpenFlow "controller", provide coherent control of the network apparatus. However, the "controlling software" of a virtual machine might be a hypervisor. There is a desire by network operators and service providers to control, configure, manage or set policy on controlling software and do so using applications for different network control and manipulation options.

Software Defined Networks infrastructure exists in most deployed networks today. Some of these components are being standardized by various organizations, as well as some being already standardized by the IETF. However, no standards or open specifications currently exist to facilitate end-to-end operation of a software defined network, specifically one that provides open APIs for applications to control the network services and functions offered by device control planes or other "controlling" software. The goal of this document is to outline the problem area of SDN for the IETF.

Section 2 discusses the use cases for SDN. Section 3 presents the SDN model and problem area to be considered by the IETF. Section 4 discusses how existing protocols can be reused to define the SDN interfaces, service discovery and object models.

1.1. Terminology

This document uses the following terms:

Control Plane: In a router or switch, the control plane is the part of the router firmware/software architecture that is in charge of the logic behind things such as constructing a map of the network topology, running network protocols or management functions and then ultimately instructing the device's data plane to realize any forwarding or switching actions that must be enabled. While conceptually separate in a logical sense, the control plane is often physically separated from the data plane of a device either by running on a processor dedicated to this function, or even run externally from the device on another device or process (i.e.: in the case of OpenFlow, centralized routing, etc...).

Data Plane: This is the part of a network device that is responsible

for the actual (i.e.: physical) forwarding and manipulation of data that comes into and is transmitted out of a device. The data plane of a device is typically very tightly bound to the specific nature of the hardware of that particular forwarding component of a device, and as such is often kept separated from the more generic control plane. An example of a data plane would be the switching fabric and port processors of a router.

Controlling Software: In the case of network devices, this is analagous to the control plane. However, in the case of virtualization technologies, this may be present in the form of a hypervisor, for example.

Application Programming Interface (API): is a particular set of rules and specifications that software programs can follow to communicate with each other. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers.

Software As a Service (SaaS): Sometimes referred to as "on-demand software," is a software delivery model in which software and its associated data are hosted by a service provider and connected back to the customer via The Internet. These are sometimes referred to as "cloud services" as well.

Managed Network Service Provider (MSP): Provides network-based connectivity and services to End Users, as well a managed service. For example, one popular MSP might offer virtualized machines (VMs) and a virtual private network (VPN) connectivity between these VMs with external connectivity to this VPN of machines via a managed business grade metro ethernet link to the customer's premise.

Communications Service Provider (CSP): A traditional "telco" service provider that offers data connectivity as a service to its customers.

1.2. SDN Background

Readers are assumed to be familiar with the architecture, features and operation of SDNs. For readers less familiar with the operation of SDNs, the following resources may be useful:

[Provide references TBD]

2. SDN Use Cases

An increasing number of MSPs are deploying SDNs in order to both offer more cost-effective service offerings, but also to reduce internal costs of managing and operating those services.

Since SDNs allow for a more consistent and shorter time-to-market model of developing management software for various network-based services, service providers are moving towards using various proprietary schemes for this. SDNs are being used to deliver various types of services that provide externally consumable SaaS offerings, as well as those used internally to manage and manipulate their network infrastructure.

Some MSPs operate over multiple geographies and couple infrastructure from different MSPs, SPs and possibly SaaS offerings. In these cases, it is important to provide the MSP that offers the ultimate service to the customer with a clean, consistent and efficient interface to all of the infrastructure it relies on. Furthermore, from their perspective, being able to unwind the "russian doll" of nested infrastructure and services that might be rolled together for their service offering in cases where trouble shooting is required for example, is paramount.

In the simplest of cases it may not seem obvious that the use of a standardized SDN infrastructure would be necessary; however, in typical medium and large data center offerings that are quite common today, the management of the physical elements is a small part of the larger puzzle for the MSP or CSP. When network elements become virtualized and are then used to construct components of services being offered, an operator can quickly multiply the number of management "devices" or more commonly "elements", by many orders of magnitude. It is here that the problem of lack of open interfaces for SDN component interconnection and discovery becomes clear.

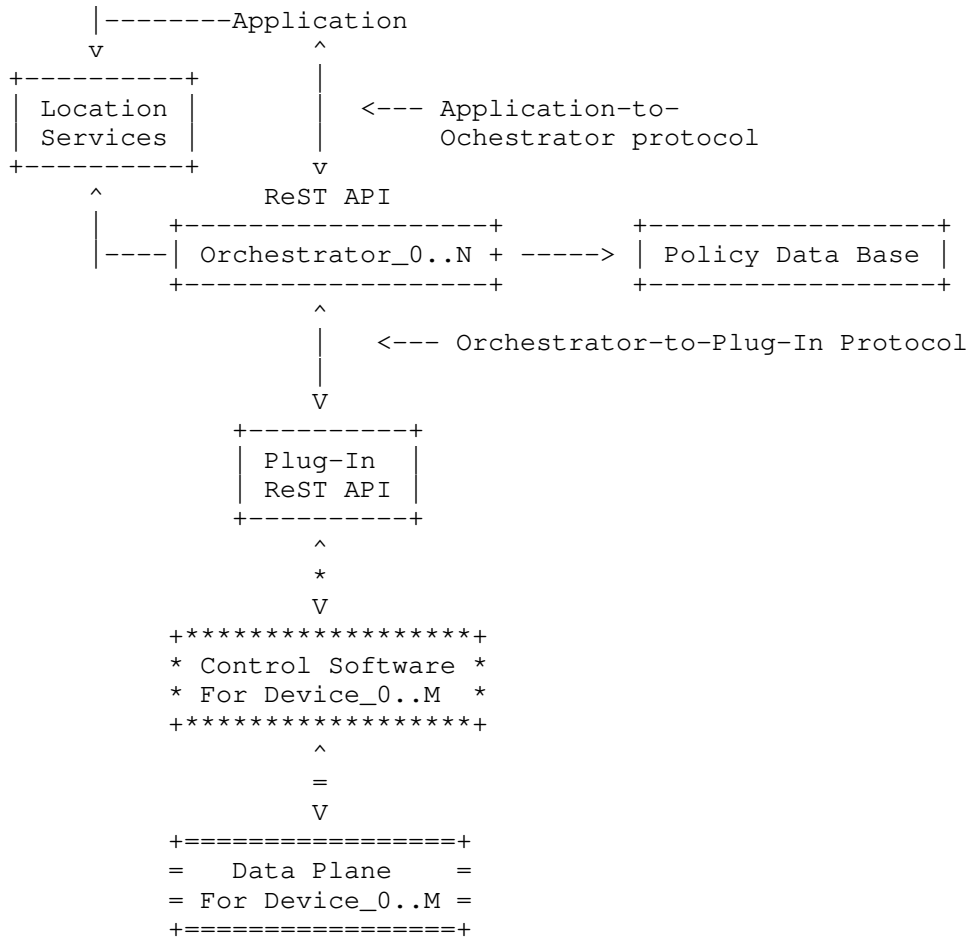
Again, for this requirement, SDN operators (over-the-top SDN operators or NSPs) are faced with a lack of open specifications and best practices.

Use cases for SDN Interconnection are further discussed in <TBD>

3. SDN Model & Problem Area for The IETF

Managing a network of deployed SDN components involves interactions among multiple different functions and components that exist within the network. Some of these components are virtual and some of these

components are real; all should be made available to be managed and manipulated, given the appropriate access, authentication, and policy hurdles have been crossed. Only some of those require standardization. The SDN model and problem area proposed for IETF work is illustrated in Figure 1. The candidate problem area (and respectively the non-goals) for IETF work are shown in Figure 2.



<--> interfaces and objects inside the scope of SDN
 +---+

<*> interfaces and objects may be within the scope of SDN
 +**+ insofar as modifications are needed to support SDN

```
<==> interfaces and objects outside the scope of SDN
+==+
```

Figure 1: SDN Problem Area

3.1. Candidate SDN Problem Area for IETF

Listed below are the the interfaces required to connect an application with the SDN components and protocols in a network. This constitutes the problem space that is proposed to be addressed by a potential SDN working group in the IETF. The use of the term "interface" is meant to encompass the protocol over which SDN data representations (e.g. SDN Metadata records) are exchanged as well as the specification of the data representations themselves (i.e. what properties/fields each record contains, its structure, etc.).

- o SDN Orchestrator-to-Application Interface: This interface allows the SDN Orchestrator or "controller" system to be interconnected with applications. This interface may support the following:
 - * Allow bootstrapping of the interface between the Orchestrator and interested applications.
 - * Allow the applications to authenticate.
 - * Allow applications to learn of which objects they have authorization to manipulate, or to interact with objects belonging to controlling software.
- o SDN Orchestrator-to-Plug-In Interface: This interface allows the SDN Orchestrator to interconnect with the controlling software of devices.
- o SDN Orchestrator-to-Policy Database Interface: This interface allows the SDN Orchestrator to interconnect with policy, authentication and authorization databases.
- o SDN Orchestrator-to-location services Interface: This interface allows the SDN Orchestrator to interconnect with location services in order to:
 - * Register itself as a local Orchestrator.
 - * Allow other Orchestrators and applications to find it.
- o SDN Orchestrator-to-SDN Orchestrator Interface: This interface allows one SDN Orchestrator to interconnect with one or more Orchestrators in order to:
 - * Form a failover/high-availability relationship
 - * distribute mappings of controlling software-to-Orchestrator

- * bootstrapping of the other SDN Orchestrators.
- * configuration of the other SDN Orchestrators.
- o SDN Logging interface: This interface allows the Logging system in interconnected SDN Orchestrators to communicate the relevant activity logs in order to allow log consuming applications to operate in multi-SDN Orchestrator environments. For example, this interface can be used to collect logs from SDN Orchestrators to provide reporting and monitoring to the M/CSP of SDN activities.

As part of the development of the SDN interfaces and solutions it will also be necessary to develop and agree on common mechanisms for how to define the object schemas used to query object model stores of each controlling software element.

4. Design Approach for Realizing the SDN APIs

This section expands on how SDN interfaces can reuse and leverage existing protocols. First the "reuse instead of reinvent" design principle is restated, then each interface is discussed individually with example candidate protocols that can be considered for reuse or leverage. This discussion is not intended to pre-empt any WG decision as to the most appropriate protocols, technologies and solutions to select to solve SDN but is intended as an illustration of the fact that the SDN interfaces need not be created in a vacuum and that reuse or leverage of existing protocols is likely possible.

4.1. Relationship to the OSI network model

The SDN interfaces called out above in Section 3.1 within the SDN problem area all operate at the application layer (Layer 7 in the OSI network model). Since it is not expected that these interfaces would exhibit unique session, transport or network requirements as compared to the many other existing applications in the Internet, it is expected that the SDN interfaces will be defined on top of existing session, transport and network protocols.

4.2. "Reuse Instead of Reinvent" Principle

Although a new application protocol could be designed specifically for SDN we assume that this is unnecessary and it is recommended that existing application protocols be reused or leveraged such as HTTP[RFC2616] to devine the SDN interfaces.

4.3. Application to SDN Orchestrator Interface

4.4. SDN Orchestrator to Plug-In Interface

4.5. SDN Logging Interface

The SDN Logging interface enables details of logs or events to be exchanged between interconnected SDN Orchestrators, where events could be:

- o Log lines related to the connection of a new Orchestrator, or disconnection of an existing one.
- o A fail-over, switch-over event. Similarly, high-availability synchronization messaging.
- o Real-time or near-real time events before, during or after SDN Orchestrator commands noting which application instructed it to perform the operation.
- o Operations and diagnostic messages.

Several protocols already exist that could potentially be used to exchange SDN Orchestrator logs including SNMP Traps or Informs, syslog, s/ftp, HTTP POST, or ReST, etc. although it is likely that some of the candidate protocols may not be well suited to meet all the requirements of SDN. For example SNMP Traps pose scalability concerns, and syslog may have potential compatability issues.

Although it is not necessary to define a new protocol for exchanging logs across the SDN Logging interface, a SDN WG would still need to specify:

- o The recommended protocol to use.
- o A default set of log fields and their syntax & semantics. Today there is no standard set of common log fields across different content delivery protocols and in some cases there is not even a standard set of log field names and values for different implementations of the same delivery protocol.
- o A default set of events that trigger logs to be generated.

4.6. SDN Orchestrator to Location Services Interface

4.7 SDN Orchestrator to Policy Database Interface

4.8 SDN Orchestrator to SDN Orchestrator Interface

5. Gap Analysis of relevant Standardization and Research Activities

There are a number of other standards bodies and industry forums that are working in areas related to SDNs, and in some cases related to SDN. This section outlines any potential overlap with the work of the SDN WG and any component that could potentially be reused by

SDN.

A number of standards bodies have produced specifications related to SDNs, namely:

- o Open Network Forum has a dedicated specification called Open Flow which specifies a relationship to an external "control plane" interfacing to a Hardware Abstraction Layer (HAL) that is implemented on Open Flow-capable hardware.

6. Relationship to relevant IETF Working Groups

6.1. ALTO

As stated in the ALTO Working Group charter [ALTO-Charter]:

"The Working Group will design and specify an Application-Layer Traffic Optimization (ALTO) service that will provide applications with information to perform better-than-random initial peer selection. ALTO services may take different approaches at balancing factors such as maximum bandwidth, minimum cross-domain traffic, lowest cost to the user, etc. The WG will consider the needs of BitTorrent, tracker-less P2P, and other applications, such as content delivery networks (SDN) and mirror selection."

In particular, the ALTO service can be used by an SDN-aware application to improve its selection of an SDN Orchestrator. For example, an application wishing to provision MPLS L3 VPNs on behalf of some virtual machines in a local data center cluster may wish to take advantage of the ALTO service in its decision for selecting a relatively close SDN Orchestrator to complete its operations.

However, the work of ALTO is complementary to and does not overlap with the work proposed in this document because the integration between ALTO and a SDN would fall under the category of using an existing protocol. One area for further study is whether additional information should be provided by an ALTO service to facilitate SDN Orchestrator selection. For example, loading or fail-over characteristics could be one consideration.

7. IANA Considerations

This document makes no request of IANA.

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

8. Security Considerations

SDNs comes with a range of security considerations such as how to enforce control of and access to objects managed by the SDN Orchestrators and making sure that in line with the M/CSP policy.

9. Acknowledgements

The authors would like to thank David Meyer, Ping Pan, Lyndon Ong, Danny McPherson for their review comments and contributions to the text.

10. References

10.1. Normative References

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

10.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[ALTO-Charter]
"IETF ALTO WG Charter
(<http://datatracker.ietf.org/wg/alto/charter/>)".

[Draft-Lee] L. Young, Bernstein, G., Kim, T., Shiimoto, K., and Dios, O.,
"Research Proposal for Cross Stratum Optimization (CSO) between Data Centers and Networks",
draft-lee-cross-stratum-optimization-datacenter-00.txt

Appendix A. Additional Material

Note to RFC Editor: This appendix is to be removed on publication as an RFC.

A.1. Non-Goals for IETF

Listed below are aspects of content delivery that the authors propose be kept outside of the scope of a potential SDN working group:

- o The interface between Controlling Software (i.e.: control plane) and the device's data plane.
- o Definition of any hardware abstraction layer (HAL)

A.2. Prioritizing the SDN Work

A.3. Related standardization activities

OpenFlow has pioneered the concept of software-defined network via a concept called a FlowVisor. It has introduced a new packet forwarding methodology to be applied on hardware or software L2 switches. OpenFlow Version 1.0 and 1.1 have been in research trials and testing in virtualized environments. The new versions will address issues such as extendibility, modularity and carrier-grade. Currently, OpenFlow does not support a mechanism to interface with network devices through the existing IP/MPLS control-plane protocols, although some work has begun to investigate this.

NETCONF/YANG provides a XML-based solution for network device configuration. It has been in wide-deployment. By definition, it supports client-to-server configuration, and server-to-client alarms or feedback (The servers are the devices/systems to be configured, the clients are the network configuration/management systems). NETCONF provides support for executing configuration change transactions over multiple devices.

ALTO is a server solution designed to gather network abstraction information and interface with applications (such as P2P) for more efficient traffic distribution. It does not require configuring the underlying network devices.

PCE is a client-server protocol that operates in MPLS networks that enables the network operators to compute and potentially provision optimal point-to-point and point-to-multipoint connections. However, PCE does not interface with applications to optimize traffic from user applications.

DMTF is a cloud computing standardization organization, which have defined many virtualization management interfaces using Restful API. However, it does not include any interface to the underlying networks.

A.4. Related Research Projects

A.4.1. IRTF Cross Stratum Optimizaiton Research Group

Some information on SDN motivations and technical motivations in the IRTF's Cross Stratum Optimization Group [Draft-Lee].

Authors' Addresses

Thomas D. Nadeau (editor)
CA Technologies, Inc.
273 Corporate Drive
Portsmouth, NH 03801
Email: thomas.nadeau@ca.com

Ping Pan
Infinera Corporation
169 Java Drive
Sunnyvale, CA 94089
Phone: (408) 572-5200
Email: ppan@infinera.com

IETF
Internet Draft

Ping Pan
(Infinera)
Lyndon Ong
(Ciena)
Shane Amante
(Level 3)

Expires: April 31, 2012

October 31, 2011

Software-Defined Network (SDN) Use Case for
Bandwidth on Demand Applications

draft-pan-sdn-bod-problem-statement-and-use-case-01.txt

Status of this Memo

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

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/lid-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 January 31, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

Bandwidth on Demand services are offered by network operators in industry and research sectors to support the needs of selected customers needing high bandwidth point-to-point connections.

Without a standard interface for controlling the use of network resources, user applications and services are subject to limits of layering, security and interoperability across multiple vendors of network equipment.

In this document, we argue the necessity in providing network information to the applications, thereby enabling the applications to directly provision network elements associated with the relevant applications.

Table of Contents

1. Introduction.....	3
2. Related Work.....	4
3. Problem Definition.....	4
4. The Role of an SDN Layer.....	6
5. Use Cases.....	7
5.1. Scheduled/ Dynamic Bandwidth On-Demand Service.....	7
5.2. Multi-Layer BoD Support.....	8
5.3. Virtualized Network Service.....	9
5.4. BoD Actions Supported by the SDN Orchestrator.....	9
6. Security Consideration.....	10
7. IANA Considerations.....	10
8. Normative References.....	10
9. Acknowledgments.....	11

1. Introduction

Bandwidth on Demand services are offered by network operators in industry and research sectors to support the needs of selected customers needing high bandwidth point-to-point connections. Such services take advantage of dynamic control of the underlying network to set up forwarding and resource allocation as requested by the customer. Some control is given directly to the customer via a portal so that there is no need to go through an intermediate stage of service order provisioning on the part of the network operator.

Currently such services are often based on management interfaces to vendor equipment that are vendor-specific, and as a result the operator must redesign its supporting control application for each vendor domain, or limit their offering to a single vendor domain.

In this document, we propose that providing a common interface to networks of different vendors and technologies would enable the network provider to offer Bandwidth on Demand and other services that are faster to deploy across a wide range of network equipment by using additional network information.

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

2. Related Work

There has been much work in this area in recent years. OpenFlow has defined an architecture for offering virtualized network control through a centralized controller and proxies called FlowVisors. These allow users to configure forwarding of packets within slices of the network partitioned off for their use. The controller is designed to control each network element directly through a dedicated control interface. It is not designed to work with existing control plane protocols.

More generally, TMF has developed models and interfaces for operations and administration of networks through the north-bound interface provided by the element management system. These interfaces are not intended for real-time control of the network element and need to take into account variations in the design and features of different types of equipment.

PCE is a client-server protocol that operates in MPLS networks that enables the network operators to compute and potentially provision optimal point-to-point and point-to-multipoint connections. However, PCE does not interface with applications to optimize traffic from user applications.

3. Problem Definition

Figure 1 illustrates the relationship between application and network today, where customer control of bandwidth on demand is provided through applications created by the network operator supporting the user interface, features and backend accounting for the service. Such applications are used in single domain deployments and have limited visibility of underlying IP/MPLS and Transport networks and, most importantly, resource availability on those networks.

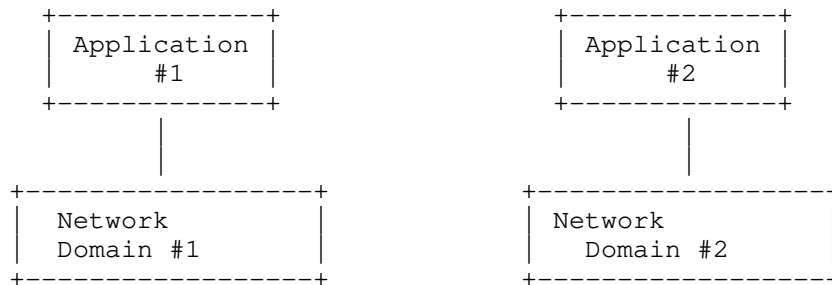


Figure 1: Application to network relationship today

This presents a number of challenges and problems. Without a standard interface to the network elements that comprise one or more network domains and their associated control software, each bandwidth on demand supporting application must be built for a specific set of vendor equipment and is not easily generalizable to different vendors or even different equipment offered by a single vendor. While signaling interfaces such as the UNI could offer standardized access to network control, such interfaces have not been adopted because they provide minimal security and functionality and are designed for more of a peer relationship between network elements, traditionally at only a single (peer) layer of the network.

Similarly, bandwidth on demand applications must be designed for a single technology, which restricts the range of use and potential users. If Domain #1 uses SDH, for example, and Domain #2 uses OTN it may be necessary to design supporting Application #2 from scratch even though Application #1 has been successfully offering service. Ideally the interface should allow some level of technology independence, as well as potentially integration to permit control of multiple layers simultaneously (esp. packet and circuit).

Third, the application is generally limited to simple services connecting a source to destination, because interfaces hide network topology and do not allow visualization of the topology for different customer views. For some services users may wish to exercise control over path routing aspects such as shared risk, required latency characteristics or inclusion or exclusion of areas for policy reasons.

4. The Role of an SDN Layer

To solve the above problem, the proposal is to introduce a software-driven network (SDN) layer (as shown in Figure 2), that is responsible for network virtualization, programmability and monitoring, between supporting applications and the network.

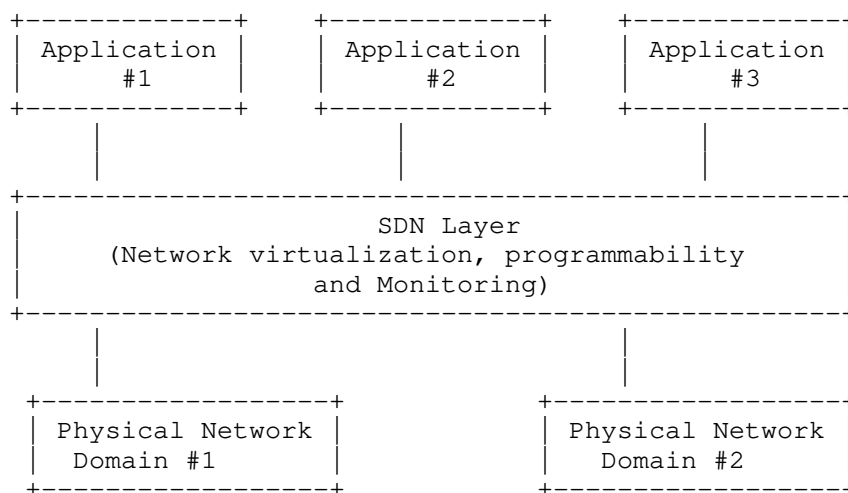


Figure 2: Application to network relationship for SDN

The purpose of the SDN Layer is to enable the applications supporting bandwidth on demand services to access information about and control (aggregate) traffic flows at various layers of the network through a standard, secure and customizable interface. Applications can visualize the traffic flows at the network layer, and manage the mapping or binding between its traffic flows from edge-to-edge through the associated networks.

The implementation of an SDN Layer involves interfacing among different types of applications and different types of network domains, based on technology or vendor, administrative or policy control. Standardized interfaces must be defined to support this.

The architecture should be agnostic as to the type of network control plan used in a supporting domain. The focus of work should

be on providing richer access to control of network resources rather than on the scheme for network control used in the domain.

5. Use Cases

5.1. Scheduled/ Dynamic Bandwidth On-Demand Service

Figure 3 illustrates the flow of a scheduled or dynamic bandwidth service. In the simplest case, connectivity may already be provided between user-specified endpoints, however the bandwidth allocated between endpoints can be varied within some overall limit based on a predefined schedule or on spontaneous customer request. Note that allowing bandwidth to be partitioned so that a scheduling application has control over some pre-allocated set of resources is necessary to support the scheduled BoD service. Also, the SDN layer ideally hides the specific technology used to support the connection, offering control of the service with associated rate, latency and recovery features.

In more sophisticated services, the customer may be allowed to create new connections within a specified set of endpoints and delete such connections when the connectivity is no longer required.

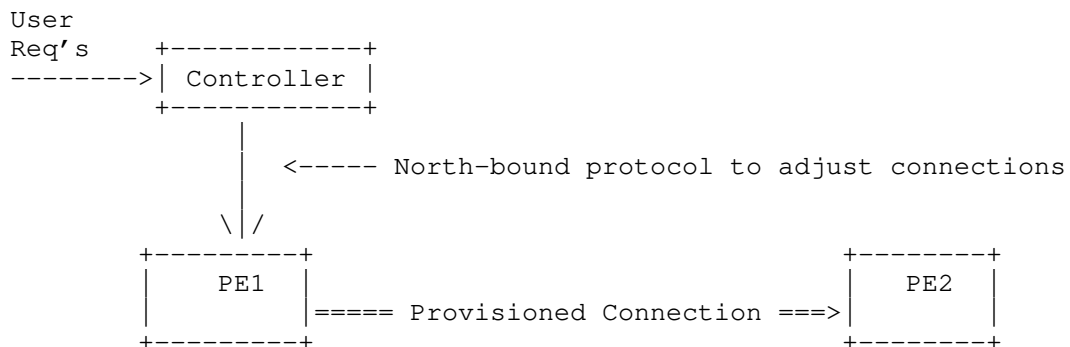


Figure 3: Scheduled/Dynamic BoD Service

5.2. Multi-Layer BoD Support

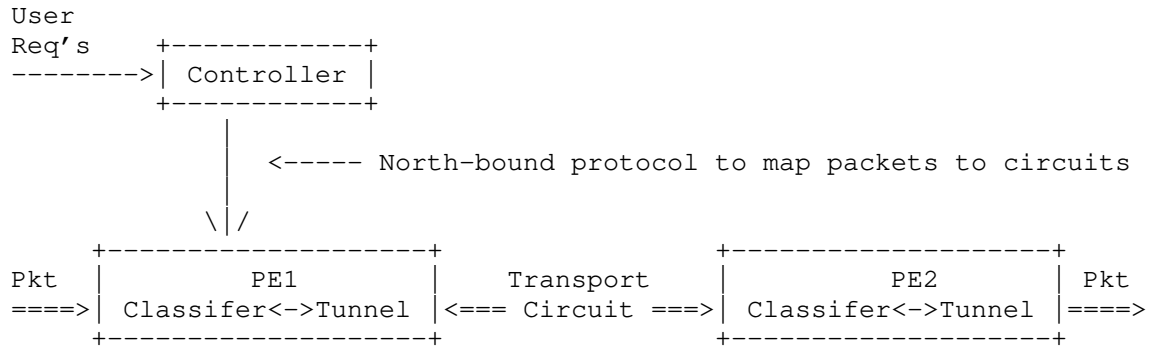


Figure 4: Multi-Layer BoD service

Figure 4 illustrates a BoD service that supports multi-layer network control. This extends allows the network operator's supporting applications to combine control of packet forwarding through guaranteed bandwidth tunnels that connect sites in a (virtual) private network as requested dynamically by the BoD customer. Different transport network technologies may be used to provide the server layer transport functions so that the application can evolve easily with new transport technologies.

5.3. Virtualized Network Service

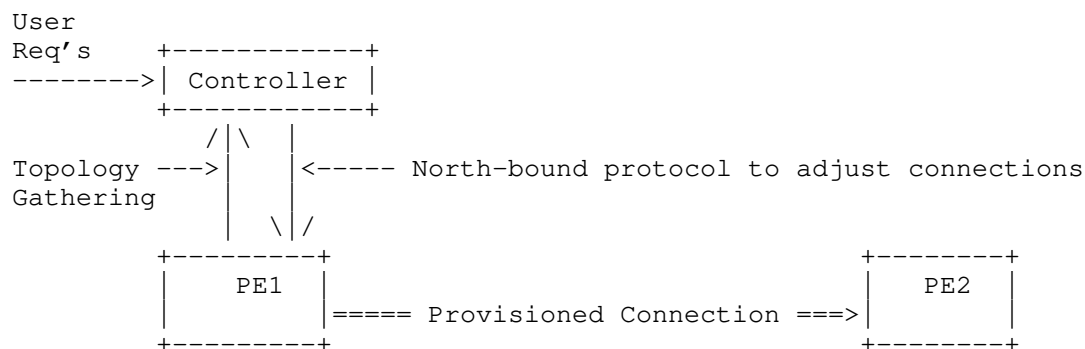


Figure 5: Virtualized network service

Figure 5 illustrates the flow of a virtualized network service that offers some degree of topology visibility and control in addition to the features of scheduled or dynamic BoD. For some customers it may be desirable to provide visibility into the topology of the resources they control, in order for the customer so they may control the physical and/or virtual topology of the resources used within their dedicated domain.

If this topology information is provided together with associated cost, latency, SRLG, etc. for the links and nodes in the topology, the customer is provided with additional flexibility to manipulate routing of their data flows so as to balance the cost, latency, energy efficiency or survivability using knowledge of client applications and their particular needs and priorities.

At this time such visibility is not possible to provide, as protocols provide either no visibility into topology or full visibility into topology. For security reasons it is likely that a supporting network operator will want to limit visibility and control to some virtualized topology using functionality provided by the SDN orchestrator.

5.4. BoD Actions Supported by the SDN Orchestrator

The following summarizes actions that would be supported by the SDN orchestrator as part of a BoD service:

- increase or decrease bandwidth on an existing path between two, or more, network clients;
- dynamically learn if resources are available, (e.g.: bandwidth, latency, SRLG, etc.) to create a path between two, or more, network clients;
- create a path and assign associated characteristics, (e.g.: bandwidth, latency, SRLG, etc.) that connect two, or more, network clients;
- configure mapping of packets, Ethernet frames, OTN frames, etc. from a client interface into a specified network path (or paths) connecting the appropriate ingress and egress client interfaces;
- configure some partition of network resources (e.g., links and link capacity connecting some set of nodes and client endpoints) to be controlled by a specific application;
- provide real or virtual topology information (links, nodes and associated information such as costs, latency, etc.) for this partition to the associated application.

6. Security Consideration

TBD

7. IANA Considerations

This document has no actions for IANA.

8. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Crocker, D. and Overell, P.(Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

9. Acknowledgments

This work is based on the conversation with many people, including Thomas Nadeau and Benson Schliesser.

Authors' Addresses

Ping Pan
Email: ppan@infinera.com

Lyndon Ong
Email: lyong@ciena.com

Shane Amante
Email: shane@level3.net

Expires: January 31, 2012

October 31, 2011

Software-Defined Network (SDN) Problem Statement and Use Cases for
Data Center Applications

draft-pan-sdn-dc-problem-statement-and-use-cases-01.txt

Status of this Memo

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

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/lid-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 January 31, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

Service providers and enterprises are increasingly offering services and applications from data centers. Subsequently, data centers originate significant amount of network traffic. Without proper network provisioning, user applications and services are subject to congestion and delay.

In this document, we argue the necessity in providing network information to the applications, and thereby enabling the applications to directly provision network edge devices and relevant applications.

Table of Contents

1. Introduction.....	3
----------------------	---

2. Related Work.....	3
3. Problem Definition.....	4
4. The Role of SDN Layer.....	6
5. Use Cases.....	7
5.1. Data Center Network Interface.....	7
5.2. Inter-data center transport.....	10
5.3. VPN.....	11
5.4. VM Mobility.....	11
6. Security Consideration.....	12
7. IANA Considerations.....	12
8. Normative References.....	12
9. Acknowledgments.....	12

1. Introduction

Service providers and enterprises are increasingly offering services and applications from data centers. Subsequently, data centers originate significant amount of network traffic. On contrast to end-to-end user applications, much of the inter-data center traffic is aggregated over a finite number of links over the backbone network. As such, without proper network provisioning, user applications and services are subject to congestion and delay.

Further, many web applications would require the interaction between multiple servers in the networks. Without adequate level of monitoring and provisioning on the network, the users may experience unacceptable services.

In this document, we argue the necessity in providing network information to the applications, and thereby enabling the applications to provision the underlying network edge devices and relevant applications directly.

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

2. Related Work

There has been much work in this area in recent years.

OpenFlow [OpenFlow] has pioneered the concept of software-defined network via FlowVisor. It has introduced a new packet forwarding methodology to be applied on hardware or software L2 switches. OpenFlow Version 1.0 and 1.1 have been in deployment in VM hypervisor environment. The new versions will address issues such as extendibility, modularity and carrier-grade. Currently, OpenFlow does not support a mechanism to interface with network devices through the existing IP/MPLS control-plane protocols.

NETCONF/YANG provides a XML-based solution for network device configuration. It has been in wide-deployment. By definition, it supports client-to-server configuration, and server-to-client alarms or feedback (The servers are the devices/systems to be configured; the clients are the network configuration/management systems). NETCONF provides support for executing configuration change transactions over multiple devices.

ALTO is a server solution designed to gather network abstraction information and interface with applications (such as P2P) for more efficient traffic distribution. It does not require configuring the underlying network devices.

PCE is a client-server protocol that operates in MPLS networks that enables the network operators to compute and potentially provision optimal point-to-point and point-to-multipoint connections. However, PCE does not interface with applications to optimize traffic from user applications.

DMTF is a cloud computing standardization organization, which have defined many virtualization management interfaces using Restful API. However, it does not include any interface to the underlying networks.

3. Problem Definition

Figure 1 illustrates the relationship between application and network today, where the applications have little or fragmented knowledge, control of or visibility of underlying networks and resources.

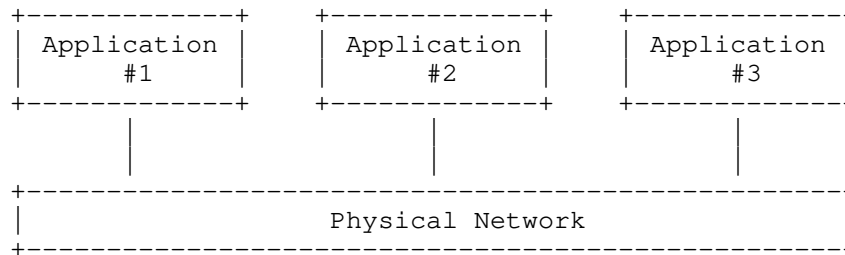


Figure 1: Application to network relationship today

This presents a number of challenges and problems.

First, due to the lack of correlation, it becomes difficult to provide service guarantees at network-level (in particular, delay) to the applications. The operators may over-provision network links to overcome to potential network congestion and packet drop within data centers. However, such practice may become too costly in many networking scenarios.

Second, many services require the interface and interaction with 3rd party back-end applications that may operate from remote locations (such as ads networks). This requires the service operators to constantly monitor the SLA conditions with remote applications, and adjust the network resources if necessary.

Third, many data center applications (such as VM) require massive user data replication on different sites for performance and redundancy purposes. Also, due to the limitation in routing and load balancing, much user traffic may be routed between data centers. As such, the inter-data center data transport need to be efficient, which requires the proper interface between applications and network.

Finally, to scale up enterprise applications on data centers, the VM's may locate on different data centers, and migrate between data centers depending on capacity and other constraints. This requires the collaboration between VM applications and the underlying networks.

4. The Role of SDN Layer

To solve the above problem, one simple way is to introduce a software-define network (SDN) layer (as shown in Figure 2), that is responsible for network virtualization, programmability and monitoring, between applications and network.

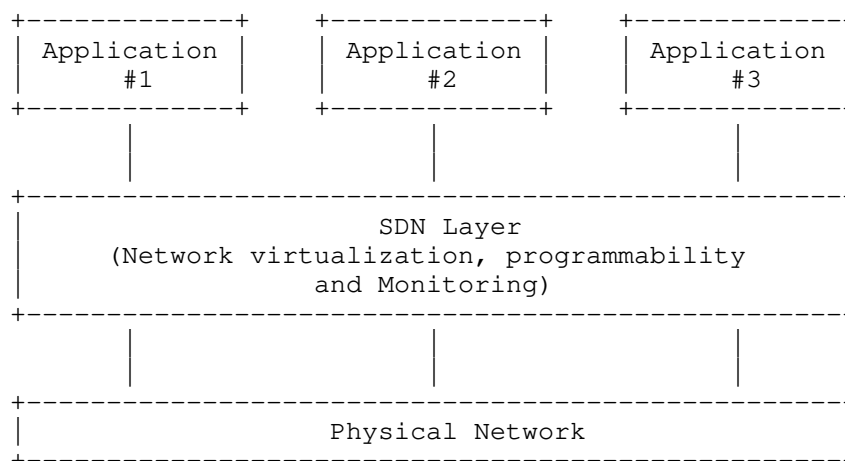


Figure 2: Application to network relationship today

The purpose of the SDN Layer is to enable the applications to visualize the traffic flows at IP network layer, and manage the mapping or binding between user traffic flows to the network connections from the edge of the networks.

There are multiple ways in implementing the SDN Layer. There have been multiple proprietary solutions in the area of interfacing Virtual Machines (VM) to the underlying network interfaces. In particular, solutions such as OpenFlow support such vision by directly programming the underlying network interface via a new protocol.

The implementation of SDN Layer involves the interfacing among applications, storage and network devices, which implies that there is a need for having a standardized interface.

Further, we recommend of utilizing the existing technologies and protocols to provision, manage and monitor network connections. The focus in realizing the SDN Layer is in optimizing the application-to-network workflow. The associated SDN protocols need to be modular, scalable and simple in design.

5. Use Cases

5.1. Data Center Network Interface

Figure 3 illustrates the data flow in data centers.

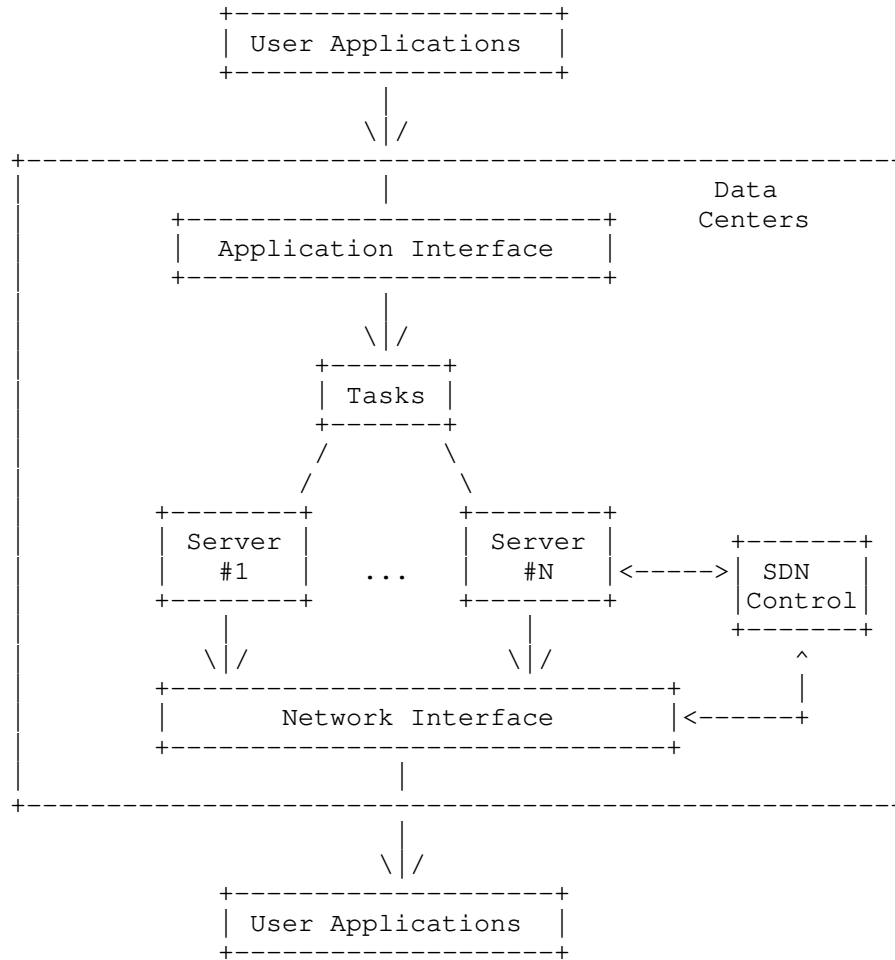


Figure 3: Data Center Traffic Flow

The data centers are designed to scale up to handle a large volume of user requests. To handle the user requests, the application interface would process and bundle the requests to different servers. Depending on the application, the data may flow between the servers or be forwarded to the users through network interface.

Note that when the servers transmit data, they typically do not have the knowledge on network connection bandwidth, delay and distance information. For intra-data center communication, this can be compensated by over-provisioning local networks. However, for transferring data between two remotely located data centers, the applications have no control of the data transmission.

Further, today, when setting up VM's over different servers, extensive manual configuration may be required. For example, all the traffic belongs to the same group/enterprise must share the same VLAN over all involved servers. This can potentially handicap the usability of the applications.

In this case, it would be desirable to have a standardized SDP protocol that can be used by the applications to interface with the networks. Through this protocol, the applications should be able to assign VLAN values to the appropriate VM sessions over all servers, and interface with the connected networks to balance the traffic load if necessary.

5.2. Inter-data center transport

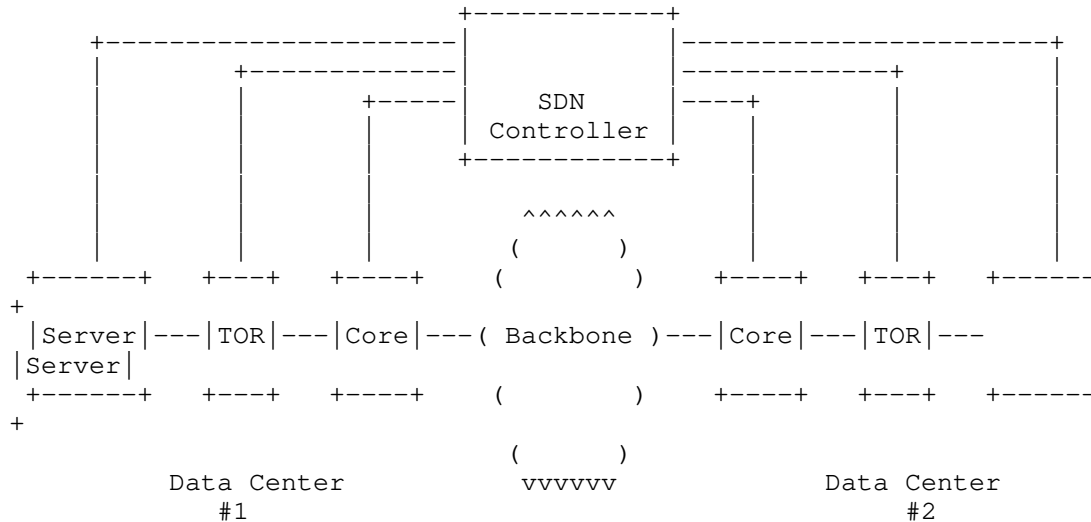


Figure 4: Inter-data center transport

When transporting data between data centers, the packets will be encapsulated into one or multiple tunnels before sending over the Internet. Traffic engineering is typically applied at tunnel-level. For instance, user IP packets at servers may be encapsulated first into a VLAN tunnel, and then aggregated into MPLS LSP's at the core node.

In this case, it would be desirable of having a SDN controller to coordinate the aggregation procedure. The controller is responsible for determining the mapping of the VLAN's to the MPLS LSP's. Further, it is possible that the controller can interface with the core node to adjust the LSP bandwidth.

6. Security Consideration

TBD

7. IANA Considerations

This document has no actions for IANA.

8. Normative References

- [1] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [2] Crocker, D. and Overell, P. (Editors), "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, Internet Mail Consortium and Demon Internet Ltd., November 1997.

9. Acknowledgments

This work is based on the conversation with many people, including Thomas Nadeau, Lydon Ong and Benson Schliesser.

Authors' Addresses

Ping Pan
Email: ppan@infinera.com

Thomas Nadeau
Email: Thomas.nadeau@ca.com

Network Working Group
Internet Draft
Intended status: Information
Expires: April 2012

D.Stiliadis
F.Balus
W. Henderickx
Alcatel-Lucent

N. Bitar
Verizon

Mircea Pisica
BT

October 31, 2011

Software Driven Networks: Use Cases and Framework

draft-stiliadis-sdn-framework-use-cases-01.txt

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/lid-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 April 31, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

This document presents an application framework and associated use cases to help define the scope of the SDNP working group. The use cases start with an abstract representation of a multi-tier application and illustrate the composition of a network service that can meet the requirements of the particular application. The service composition process is split into several steps, including application requirement specification, network mapping, service binding, and policy control. We also provide examples of interactions between an SDNP controller and L2 and L3 control layer protocols in order to deliver the end-to-end service. Finally, as a derivate of that, an architecture framework for SDNP that meets these requirements is proposed.

Table of Contents

1. Introduction.....	3
2. General Terminology.....	4
3. Framework.....	4
3.1. Application Definition.....	5
3.2. Network Mapping.....	7
3.3. Network Service Binding.....	8
3.4. Policy Definition.....	9
4. Examples of Service Binding.....	10
4.1. Example: An end-to-end data center service.....	10
4.1.1. LAN Configuration.....	11
4.1.2. IPVPN Configuration.....	12
4.2. Configuration Alternatives.....	12
4.2.1. Network Element based Configuration.....	12
4.2.2. Network Management based Configuration.....	13
5. SDN Model and Reference Architecture.....	13
5.1. Scope and Approach.....	15
6. Security Considerations.....	15
7. IANA Considerations.....	15
8. Conclusions.....	16
9. References.....	16
9.1. Normative References.....	16
9.2. Informative References.....	16
10. Acknowledgments.....	16

1. Introduction

In order to define the scope of the proposed SDNP working group, we present a generic application use case and a sequence of steps needed for mapping the application requirements to a deployable network service. We also present a generic architecture framework that can meet such requirements.

The goal of SDNP is to define a method where applications can request services from the network and these services can be automatically deployed. We view technologies such as FORCES and OpenFlow as complementary and orthogonal to SDNP. Unlike Openflow, where the goal is to introduce a disaggregated control layer, the goal of SDNP is to enable existing control planes to become more adaptable to application requirements and to allow rapid and reliable configuration changes by introducing a framework for communication between applications and network control planes. More importantly, the framework should be applicable to communicating application requirements even when a variety of network technologies is used to provide the required services. In this context, SDNP is also useful to OpenFlow type networks, since it provides the interface between applications and control planes implemented in an Openflow controller.

In order to achieve these goals, applications should be able to specify their requirements in a generic way and these requirements must be translated in an actual network service. In general, application developers do not know in advance the type of networks that will be used and they would require an abstract and flexible framework for defining their requirements.

Often times a service spans multiple administrative domains where given application requirements are met by different networking technologies. For example, providing network isolation for the application servers can be achieved by VLANs, MPLS or other underlying L2 or L3 technologies. It is possible that an application that is distributed between multiple data centers and networks be deployed through VLAN isolation in one domain and MPLS isolation in another domain, and an IP VPN in between. The application does not care as to which underlying technology is used to implement the isolation, as long as the properties of the isolation are guaranteed.

In most instances the types of network technologies that should be used to deliver a given application will depend on policies either set by the network service provider, cloud service provider, another

administrative authority, and/or the application itself. In multi-tenant environments there can be a hierarchy of policy objectives set by different organizations and the implementation of the network service must take into account all policy restrictions. For example, an enterprise IT organization can define that any application deployed by their employees must adhere to specific security requirements, such as encrypted tunnels between application servers. At the same time, because of an SLA contract, the service provider requires that the service is always deployed over redundant paths. In these cases, the requirements imposed by the application will be mapped in such a way that they comply with both policies. If for example a user within that organization tries to deploy an application with guaranteed bandwidth between servers, because of the policies defined above, the service instantiation will be over an encrypted and redundant path that satisfies the bandwidth constraint. In the next sections we provide an overview of this application driven framework and illustrate with specific examples how an SDNP controller can interact with control layer protocols.

2. General Terminology

Network Spec: The definition of the network service requirements by an application.

Network Mapping: The transformation of application requirements to a network service on specific network technologies.

Binding: The binding of a network service to specific network elements.

SDNP Controller: The software controller that allows the specification of an application Network Spec, and implements the network mapping and binding functions. In binding the network mapping to a network element, the SDN control may talk to the network element directly or via another controller such as an Element Management System (EMS) or an open flow controller.

3. Framework

First we illustrate a generic framework for mapping application requirements to network services and then we illustrate in more detail the functionality of each component. Note that this framework is just for illustration purposes and specific implementations can combine multiple functional blocks in a variety of ways.

The basic blocks are illustrated in the next Figure:

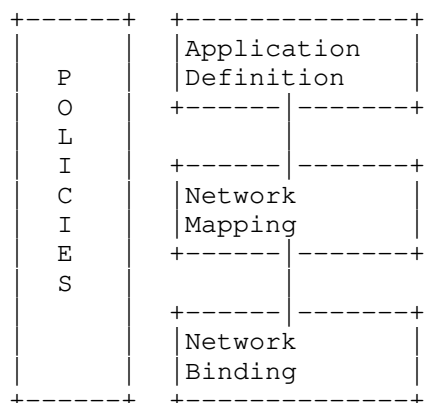


Figure 1 Generic Framework

- o Application Definition: This is a generic representation of the requirements of an application without specifying the underlying technology.
- o Network Mapping: This function translates the requirements of the application to an actual network service that can be implemented by a specific network technology.
- o Binding: This function maps the abstract network service on control planes and specific network elements.
- o Policies: Provides the repository of policies that will drive the translation between generic requirements and network technologies as well as the binding to specific elements.

3.1. Application Definition

An example of multi-tier application definition is shown in the next Figure:

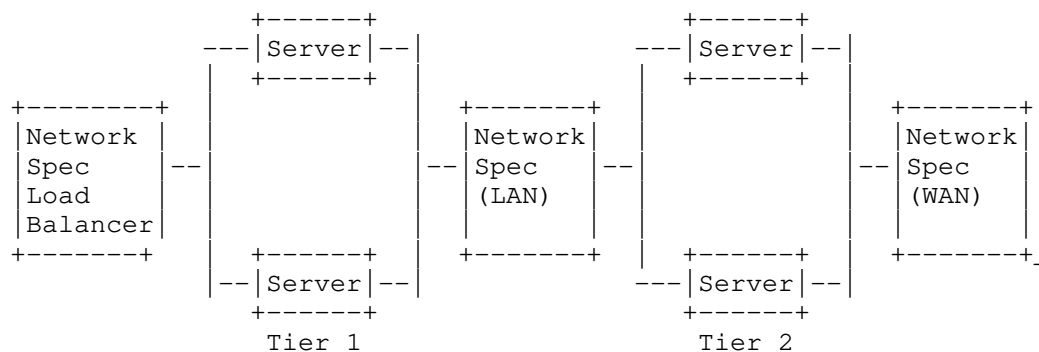


Figure 2 Multi-tier application description

From the application perspective, the definition consists of a set of compute and storage tiers interconnected by a network segment that meets specific requirements. The application does not care about the type of underlying technology, but rather about the properties of the network segment. Generic application requirements can include isolation, bandwidth, communication based on criteria other than shortest path, network redundancy, access control, etc.

In the Example of Figure 2, the application might require that a load balancer distribute load among all the Tier 1 servers. The application does not care about how the load balancer is instantiated or how it is configured, or whether it is a physical or virtual machine. If the first Tier represents a set of Web servers and the second Tier a set of Business Logic servers, the network spec between the Tiers only defines that web-servers communicate with business logic servers over a specific protocol and port and require isolation. It does not define how this isolation is achieved. The network mapping function, depending on the technology chosen will implicitly identify the need for access lists or firewall rules between the Tiers that will prevent any unauthorized access.

A different network spec will define the back-end connectivity requirements between the business logic tier and other enterprise services. For example, it might require that the Business Logic Tier must communicate with the enterprise Data Center over a secure connection, since critical data must be physically protected and stored within the enterprise premises. The application developer does not necessarily need to know whether the application is deployed across one or multiple DCs or what is the level of security required. However, the IT policies that have been supplied to the cloud provider should describe the necessary security mechanisms that must be deployed in order to comply with legal compliance rules or other policies. The policy will define for example that all access to data base Tiers must be encrypted, and the encryption strength that must be used.

Another type of network spec might define that all servers in a given Tier belong to the same LAN, since the service will rely on virtual machine motion for balancing the load or achieving reliability. In another example, machines within a Tier need to form a cluster that must support fencing that is achieved by a majority protocol between the servers. In this case, healthy servers can shutdown network connectivity to failing servers and the network must provide the necessary APIs to achieve that. At the same time, the network APIs must prevent an application belonging to one

customer to affect network connectivity of another customer or another application.

In all the above examples, the application spec defines the properties and attributes of communication between components without necessarily defining the mechanism for achieving this communication.

3.2. Network Mapping

The first transformation needed is to map the application requirements to a set of network technologies that are supported by the underlying physical network. Different service providers and networks can choose different technologies for this implementation.

For example, if the underlying network utilizes VLANs and VPLS, then it could map each of the individual networks in a different VLAN, and it could utilize VPLS for interconnecting data center VLANs with enterprise VLANs. Alternatively, if the service provider utilizes some L2 over L3 technology such as proposed in VXLAN [DRAFT-VXLAN] or PBB tunneling over IP, and IPVPN, it could use a combination of these technologies to map the application requirements to a set of network primitives.

This transformation function takes as input the application requirements, the available network services, and the policy definitions, and creates a design of a composite network service that meets the application requirements. For the example instantiation of the 2-tier application using VLANs, the resulting network service is shown in Figure 3.

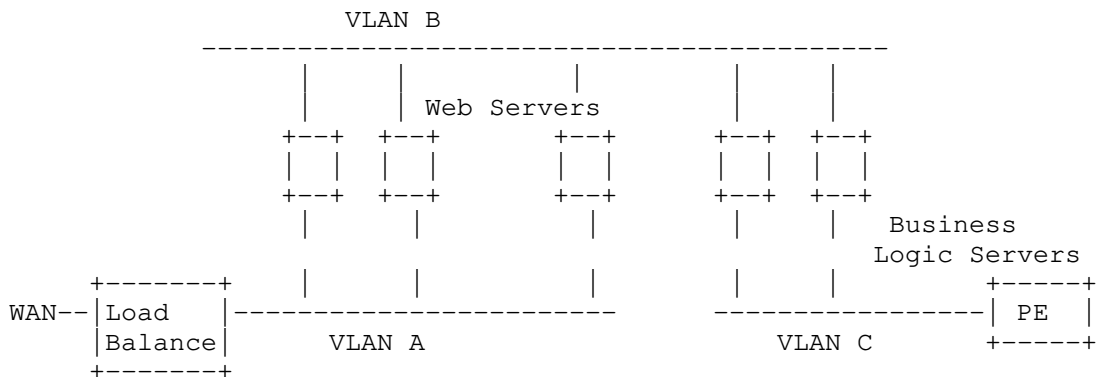


Figure 3 Implementation of 2-tier service with VLANs

In this network mapping, the DC/cloud service provider has chosen to implement the load balancer as a virtual machine, and has chosen to interconnect application Tiers through VLANs. It has chosen an IPVPN connectivity to the enterprise back-end. At this stage, although the network service has been defined, the exact VLAN numbers or nodes implementing the services have not been identified yet.

Note, that when a service spans multiple administrative domains, it is possible that different technologies are used in each domain to meet the same application requirements. For example, an application that is split between a service provider data center and an enterprise data center might rely on IEEE SPB for layer-2 isolation within the service provider DC and VLAN based isolation within the enterprise data center. In another example, a WAN request for a guaranteed bandwidth path between data centers in different domains can be implemented as a wavelength service in one domain and as an MPLS service in another domain. Therefore, the service mapping must not only define how the network service is provided in each of the domains, but it must also identify the necessary mechanisms needed for stitching services between domains. Obviously, this might introduce a very large number of permutations, and therefore the SDNP work must concentrate on frameworks and specific use cases to limit the scope.

Note also that in multi-domain implementations, there is no single master SDN controller. Each administrative domain will have its own SDN controller. In these cases, communication between SDN controllers must be done at the abstract level of service requirements rather than by defining explicit network technologies. In this way, different administrative domains can seamlessly interface to serve such applications, even when they rely on different network technologies.

The next step in the process is the binding of the network service to specific network elements.

3.3. Network Service Binding

During this step of the operation, the composite network service must be mapped to particular network elements and topologies. At this level, the necessary resources (servers, virtual load balancers, virtual firewalls, etc.) are mapped to specific physical resources. The network service interconnecting these resources must be instantiated through a sequence of programming steps between the SDN controller and the individual physical or virtual network elements or network management systems.

There are several different methods that can be used to implement the binding process and a set of different protocols that can be used for communicating with the individual network elements or network management systems. Existing protocols include SNMP, Netconf, and even Command Line Interface (CLI). Alternative solutions might rely on network management tools and specific APIs that they expose. As Openflow matures, an SDNP controller could also program services in networks that utilize Openflow by communicating with the Openflow controller.

The binding process will usually require a series of steps that can include:

- o L2 and L3 topology discovery,
- o Path selection for each service
- o Traffic engineering for providing some form of SLAs.
- o Configuration of network elements for L2 and L3 services

Depending on the underlying technologies, some of the above steps can be omitted and can be performed with distributed control planes. For example establishing an MPLS path for communicating between data centers can utilize LDP or RSVP-TE and does not need explicit knowledge of the network topology or configuring every element in the path. On the other configuring VLANs on an Ethernet network might require explicit configuration of network elements.

Nevertheless, the important characteristic of the binding process is that it requires a method for interacting with control layer protocols or network and element management systems. Even though a set of such mechanisms could be available in networks today, it is unclear that the interfaces are powerful or generic enough to allow such dynamic programming and this is the main focus of the SDNP work. Bridging the gap between the controlled environment of today's networks and an application driven network configuration will require a set of access control mechanisms that will protect the underlying infrastructure.

3.4. Policy Definition

Every transformation step in the above description must be driven by policies that are administered either by the service provider, the IT organization requesting the service or the applications themselves.

It is such policies that will determine how network requirements are mapped to network design and how services are bound to specific network elements. The policy complexity will depend on the service

offered by a provider and can include security, billing, compliance, performance related policies, etc.

4. Examples of Service Binding

As it is evident by the above description a complete framework for application-driven network programming offers several layers of abstraction and the work around SDNP must carefully choose the layer of scope in order to solve specific problems. Addressing the whole architecture might prove a huge and challenging task that will limit the usefulness or impact timely completion of the work.

Clearly SDNP work must address the binding step that will enable configuration of network elements and control layer protocols in order to deliver the service. SDNP will also need to develop the interfaces between different administrative domains for services that span multiple domains. In the next Section we present such a control protocol configuration that will illustrate the scope of control plane programmability of SDNP.

4.1. Example: An end-to-end data center service

We consider the example presented in the previous Sections. We assume that the implementation is done over an IEEE 802.1aq SPBM network within the data center and an IPVPN service will connect the DC servers with other enterprise resources. The SDNP controller will utilize two different mechanisms to program the DC LAN and IPVPN services. The logical service mapping is illustrated in the following Figure 4, and it is similar to that of Figure 3, where each VLAN is now replaced by a different service ID (ISID):

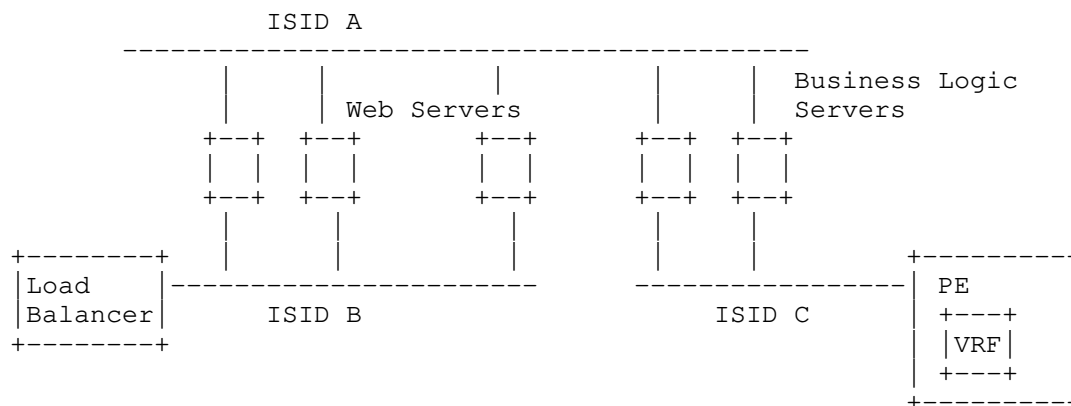


Figure 4 Service instantiation based on SPBM

4.1.1. LAN Configuration

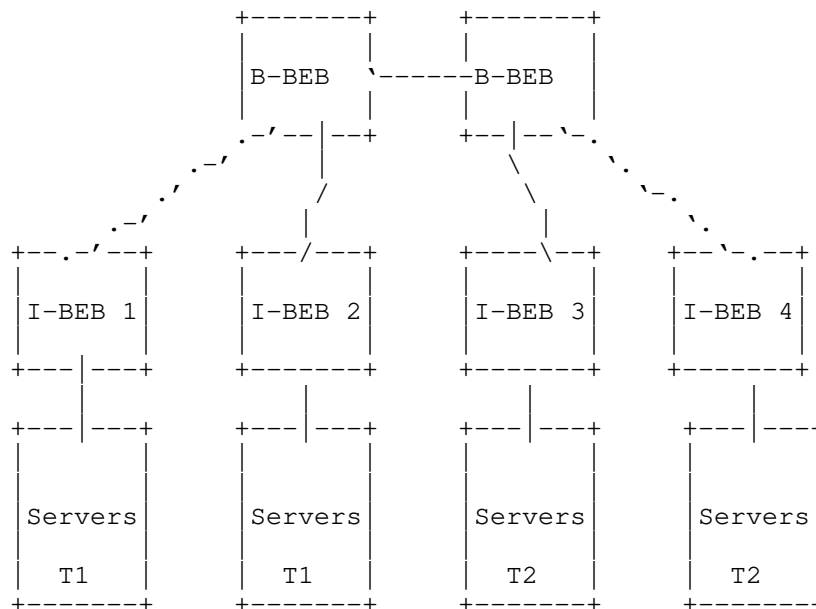


Figure 5 SPBM based DC networks

We consider a DC network based on SPBM [IEEE802.1aq] (see Figure 5). Within the DC LAN, network isolation is provided by assigning virtual machines or servers of a given data center tenant to different Service IDs (ISIDs). A set of edge bridges (I-BEBs) encapsulate Ethernet frames using PBB encapsulation. The I-BEB function can be implemented either at the Top-of-Rack switch or at the hypervisor virtual switch. A set of B-Component bridges (B-BEB) interconnect the edge bridges. The I-BEBs will map traffic from each tenant to a particular service instance (I-SID/B-VID) depending on the service requirements. In the example of Figure 5, application tier T1 is associated with I-BEBs 1 and 2 and application tier T2 is associated with I-BEBs 3 and 4. The SPBM protocol provides the mechanisms for shortest path forwarding and learning in the backbone space without the need of deploying spanning trees or MAC learning. SPBM also provides the mechanisms for service auto-discovery and flood containment when a service covers a subset of service nodes. For example, if servers of a given Tier are associated with a subset of the I-BEBs, SPBM will create the multicast tree for flooding that will be associated with only this subset of I-BEBs. In order to instantiate a VM in a new node, the corresponding I-BEB needs to be provisioned, and once this provisioning is complete SPBM will expand

the multicast tree and include this I-BEB in the multicast distribution automatically. In the example of Figure 5, if a new VM from tier T1 is associated with I-BEB 4, the multicast tree associated with this I-SID/B-VID will be expanded to cover I-BEB4.

4.1.2. IPVPN Configuration

As shown in Figure 4, the enterprise service in the DC is connected to other enterprise data centers and services through an IPVPN [RFC 4364]. The first time that a business logic tier server is instantiated and ISID C is created, the SDNP controller must also provision the PE with the corresponding VRF. The SDNP controller provides the necessary information to the PE (incoming interface, ISID/B-VID pair, route targets and route distinguishers), and instructs the PE to instantiate the VRF. Once the VRF is instantiated, the existing routing protocol mechanisms (MP-BGP) will be used to propagate the routes to the other VRFs of the same IPVPN. Note, that in this case the SDNP controller does not define forwarding behavior, but it rather instantiates part of the control plane.

The scope of the SDNP configuration is limited to the edge PE and does not include any other provisioning or configuration in other routers in the network. This provides a simple interaction between the DC management system and network protocols, since it does not impose the requirement for the DC management system to understand the full network topology.

In more complex environments with separate DC and WAN PEs, this configuration might apply only to the DC PE, and the inter-connection between DC and WAN PEs can be done using any of the options of [RFC 4364].

4.2. Configuration Alternatives

The SDNP Controller can choose one of two methods for provisioning the necessary network functions.

4.2.1. Network Element based Configuration

In the above examples, the function of the SDNP controller is limited to provisioning the "edge" elements (I-BEB, PE) and a layer-2 or layer-3 protocol propagates the necessary information through the network (SPBM or MP-B respectively). In both cases, the SDNP controller has the ability to directly program functions in the corresponding network elements and does not need access to all network elements.

In addition to the programming interface, the SDNP controller must have an exact representation of the physical topology so that it can identify which I-BEB and PE must be provisioned (i.e. it is aware of the physical connectivity between I-BEBs and servers, L2 topology and PEs). The controller can discover this information by polling the I-BEBs and/or using LLDP between I-BEBs and servers. The controller can also listen to the SPBM protocol to capture topology information. Alternatively I-BEBs can proactively notify the SDNP controller about topology changes and the activation of new physical or virtual servers.

4.2.2. Network Management based Configuration

In an alternative implementation, the SDNP controller does not directly provision network elements, but it utilizes a network management tool that is already deployed in the network. Note, that in several cases, cloud and network operators have deployed network management and OSS systems and would want all operations to be driven by these systems.

In this instance, the SDNP controller does not need to maintain detailed topology information, and it just talks to the corresponding network management system to issue the requests for I-BEB and PE provisioning.

5. SDN Model and Reference Architecture

Based on the discussion above, and using the simple example of Section 4 as guidance, we can develop the SDN reference architecture that can capture these requirements. Figure 6 outlines this architecture.

In this model, SDN Controllers expose an abstract API to applications, where they can request specific network properties such as bandwidth assignments, network isolation, routing properties, redundancy properties, security requirements, etc. Communication between different SDN Controllers enables these entities to provide services across network administrative domains without being constrained by underlying technologies. The interface is limited to communicating the requirements of the application. Each SDN Controller can translate application requirements to different technologies based on the underlying network implementations. This approach provides the maximum portability for applications and does not burden application developers with network technology specifics.

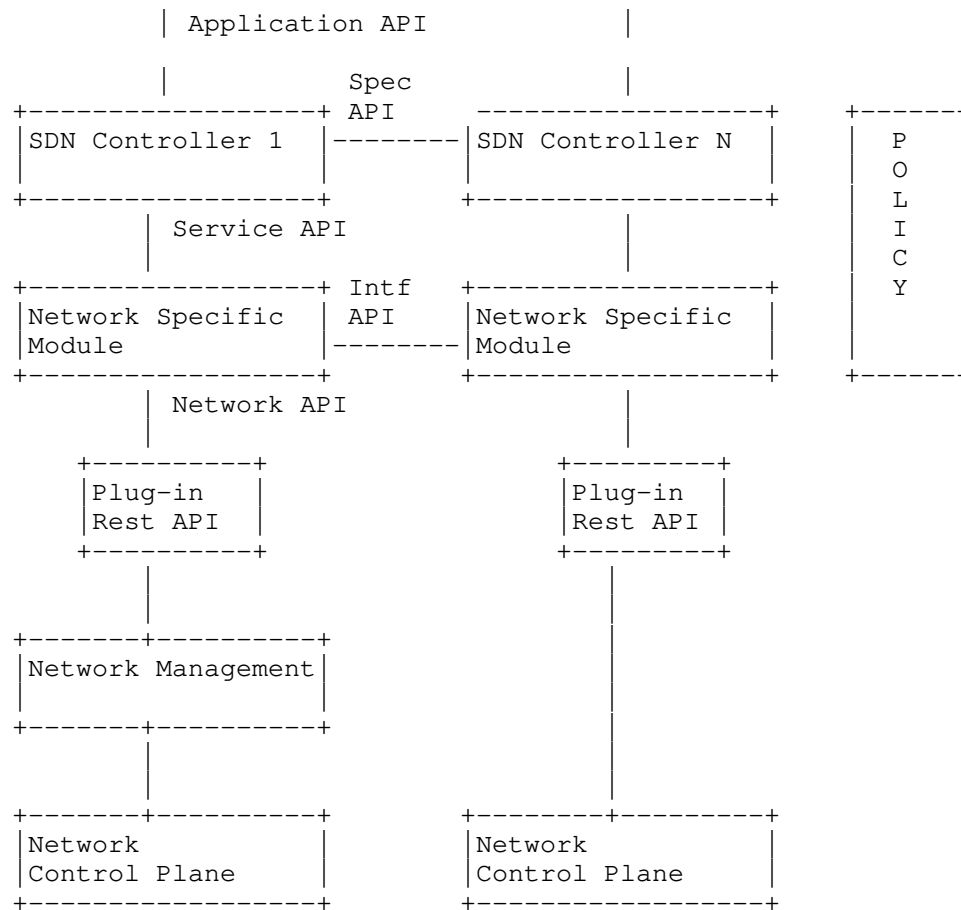


Figure 6 SNDP reference architecture

As it relates to the framework of Figure 1, the SDN controller implements the Network Mapping function and it includes the policy specification functions. The SDN controllers utilize a service API to communicate these requests with a network technology specific module that will be responsible for the translation of the requirements to specific technology primitives. An Interface API will enable network specific modules to stitch services together. This is a technology dependent API and it will enable functions such as mapping a VLAN to VPLS domain, or mapping an LSP to a wavelength and so on.

The network specific module is essentially implementing the network binding function of Figure 1. For every network technology, there is

a specific API to the control plane of network elements that will allow the binding and instantiation of the network service. Note, that in several environments it is possible that the network mapping is communicated with a pre-existing network management system and not directly with the control plane of network elements. This not only allows the evolution from current operational models, but it also enables the concurrent support of existing and future operational models. If for example an OSS is currently driving the deployment of network services through a network management system, such an approach would allow new applications to utilize the same infrastructure and provide an evolutionary path to a software defined network.

Note, that the network specific module might also include functionality related to topology discovery. The type of topology information required will depend on the underlying technology and it can range from a full network map in an Openflow environment to a limited resource view when the final binding operation is performed by a network management system.

5.1. Scope and Approach

Given the number of different networking technologies and implementations, defining all possible APIs within the scope of a single working group will be hard to tackle. The SDNP work should therefore concentrate on designing the framework, application network requirements schema, and API control and communication architecture, as well as defining the reference schema and API around a single networking technology as an example use case. Once the framework is in place, the technology dependent functions can utilize it to develop their own specific APIs and this can be done across multiple working groups.

6. Security Considerations

The SDNP controller security aspects must be addressed, including functions such as role based authentication and security of intra-SDNP controller communications.

7. IANA Considerations

IANA does not need to take any action for this draft.

8. Conclusions

This document has introduced a generic framework for mapping application requirements to specific network services within the context of Software Driven Networks. The document also outlined a reference framework as input to the definition of the scope of the SDNP work.

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.

9.2. Informative References

[IEEE802.1aq] IEEE 802.1aq Shortest Path Bridging

[RFC 4364] E. Rosen et.al, BGP/MPLS IP Virtual Private Networks, RFC 4364, February 2006.

[DRAFT-VXLAN] Mhalingham et.al., "VXLAN: A framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", draft-mahalingham-dutt-dcops-vxlan-00.txt, September 2011.

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot. We also want to thank T. Sridhar for early comments on the draft.

Authors' Addresses

Dimitri Stiliadis
Alcatel-Lucent
777 E. Middlefield Rd
Mountain View, CA 94043

Email: dimitri.stiliadis@alcatel-lucent.com

Florin Balus
Alcatel-Lucent
777 E. Middlefield Rd
Mountain View, CA 94043

Email: florin.balus@alcatel-lucent.com

Wim Henderickx
Copernicuslaan 50
2018 Antwerp, Belgium

Email: wim.henderickx@alcatel-lucent.be

Nabil Bitar
Verizon
40 Sylvan Road
Waltham, MA 02145

E-mail: nabil.bitar@verizon.com

Mircea Pisica
BT
Telecomlaan 9
Brussels 1831, Belgium

Email: mircea.pisica@bt.com

