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 as well 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, specificlly 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.

Copyright Notice

Table of Contents

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, mange 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 as well 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, specificlly 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 seperated 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 effient 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.

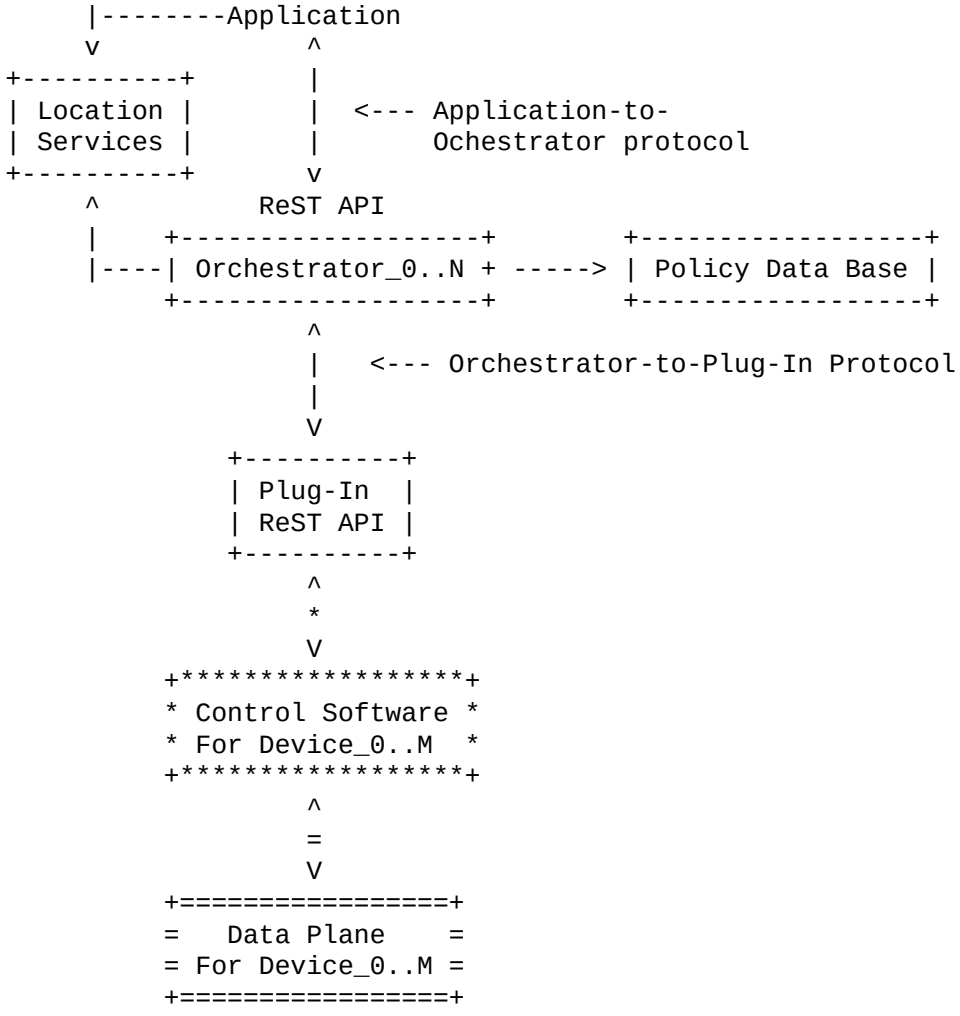
Again, for this requirement, SDN operators (over-the-top SDN operators or NSPs) are faced with a lack of open specifications and best practices.

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 modifcations 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.).

- 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 beloning to controlling software.

- SDN Orchestratort-to-Plug-In Interface: This interface allows the SDN Orchestrator to interconnect with the controlling software of devices.

- SDN Orchestratort-to-Policy Database Interface: This interface allows the SDN Orchestrator to interconnect with policy, authentication and authorization databases.

- SDN Orchestratort-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.

- 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.

- 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 inetrface 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:

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

- The recommended protocol to use.
- 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.
- 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:

- 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 with 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 characterists 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 McPhereson 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., Shiomoto, 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:

- The interface between Controlling Software (i.e.: control plane) and the device's data plane.
- 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