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 will expire on April 30, 2012.

Copyright Notice

Table of Contents

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 resorces, network resources such as router control planes, and policy and security engines. Furthermore, high availability and resilliency 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.

```
     |--------Application
     v             ^
+----------+       |
| Location |       |  <--- Application-to-
| Services |       |       Ochestrator protocol
+----------+       v
     ^          ReST API
     |    +-------------------+        +------------------+
     |----| Orchestrator_0..N + -----> | Policy Data Base |
          +-------------------+        +------------------+
                   ^
                   |   <--- Orchestrator-to-Plug-In Protocol
                   |
                   V
              +----------+
              | Plug-In  |
              | ReST API |
              +----------+
                   ^
                   *
                   V
          +*****************+
          * Control Software *
          * For Device_0..M  *
          +*****************+
                   ^
                   =
                   V
          +=================+
          =   Data Plane    =
          = For Device_0..M =
          +=================+

  <-->  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

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 Orchesetrator

The controler 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 communcations bewteen 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 Orchstrator will be designed and architected so as to exist in a distributed manner if so desired operationally. Furthermore, resilliency 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 controling 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 beloning 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.

- SDN Domain - the full domain name of the origin server
- IP address - the IPv4 (and IPv6 if available) address of the client application or SDN Orchestrator making the request
- Transaction Time - the ending time of the transfer
- 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 bewteen 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:

-

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