IJARSCT
ISSN (Online) 2581-9429
Impact Factor: 6.252
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, April 2022

A Study on Opendaylight and its Architecture

Priyanka Hadpad
Department of Information Technology
S. S. & L. S. Patkar College of Arts & Science & V. P. Varde College of Commerce & Economics, Mumbai
priyankahadpad4@gmail.com

Abstract: *OpenDaylight (ODL) is collaborative project under the Linux Foundation, provides an open-source, modular, and flexible SDN and Network Function Visualization (NFV). ODL supports a layered architecture. The southbound interface to ODL ensures that networking technologies and hardware from diverse vendors can be leveraged using ODL. The northbound interface provides APIs for end user and other technologies such as OpenStak. This report captures detail description of ODL architecture in. simplified way. In ODL north bound APIs contain topology manager, host tracker, flow programmer, static routing, and so on. In southbound, multiple protocols are supported as plugins, e.g. OpenFlow 1.0, OpenFlow 1.3, BGP-LS and SAL (Service Abstraction Layer) figures out how to fulfill the requested service irrespective of the underlying protocol used between the controller and the network devices.*

Keywords: OpenDaylight, SDN, OpenStak, OpenFlow, SAL.

I. Introduction

OpenDaylight (ODL) [1], hosted by the Linux Foundation, is an open-source platform for network programmability aimed at enhancing software-defined networking (SDN).The OpenDaylight controller is JVM software and can be run from any operating system and hardware as long as it supports Java. The controller is an implementation of the Software Defined Network (SDN) and continues to grow. And it accepted to accelerate the adoption of SDN and network Functions Virtualization (NFV), OpenDaylight provides an open platform for network programmability and cloud computing designed to enable SDN and create a solid NFV foundation for all sizes of networks.

In this application development, a set of loosely coupled modules can be integrated into one large application. “Loosely coupled” means modules are both independent and can communicate with one another. ODL architecture is developed based on the Open Services Gateway Initiative (OSGi), a dynamic module system for java. It helps for modular application development. ODL architecture is formed in a layered structure layer on the top, the platform controller layer in the middle, and the network elements represent the lower layer. The ODL’s heart is the middle layer which contains: the basic network functions such as topology, statistics, and forwarding services; the platform network functions which include modules for specific networking tasks; as well as the service abstraction layer which represents a service abstract level between the lower layer and the upper layer and it also routes service between requests layer’s modules.

II. Software Defined Networking (SDN)

Open software-defined networking (Open SDN) - the use of standards-based protocols and open interfaces to abstract the network control plane from the data plane. Open SDN enables unified control and programmability across a network of heterogeneous physical and virtual networking devices [2].

Open SDN is based on a three-tier architecture of virtual devices.

- Northbound open APIs for application developers
- An open-core controller
- Southbound standards-based data plane communication protocols

A secured southbound (SB) interface is established between each element of the underlying entire network and the controller, through which the rules are for-warded. Networking elements store the rules in a chain of flow tables and when no entry matches the packet header fields of a received packet in the flow tables, routers or switches send it to the controller. If a matching rule is found, the defined action (drop, forward to a certain port) is applied. If no matches found, the packet can be either dropped or sent to the controller[2].

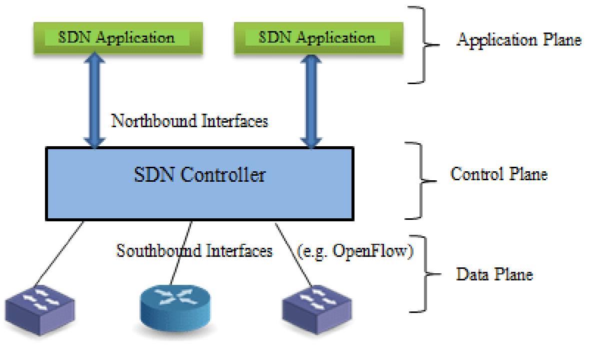

Figure 1: Software Defined Networking (SDN) Architecture

III. Architecture of OpenDaylight

OpenDaylight Helium components include a fully pluggable controller, interfaces, protocol plug-ins, and applications. The Helium controller consists of three key blocks [4]:

- The OpenDaylight controller platform
- Northbound applications and services

Southbound plugins and protocols

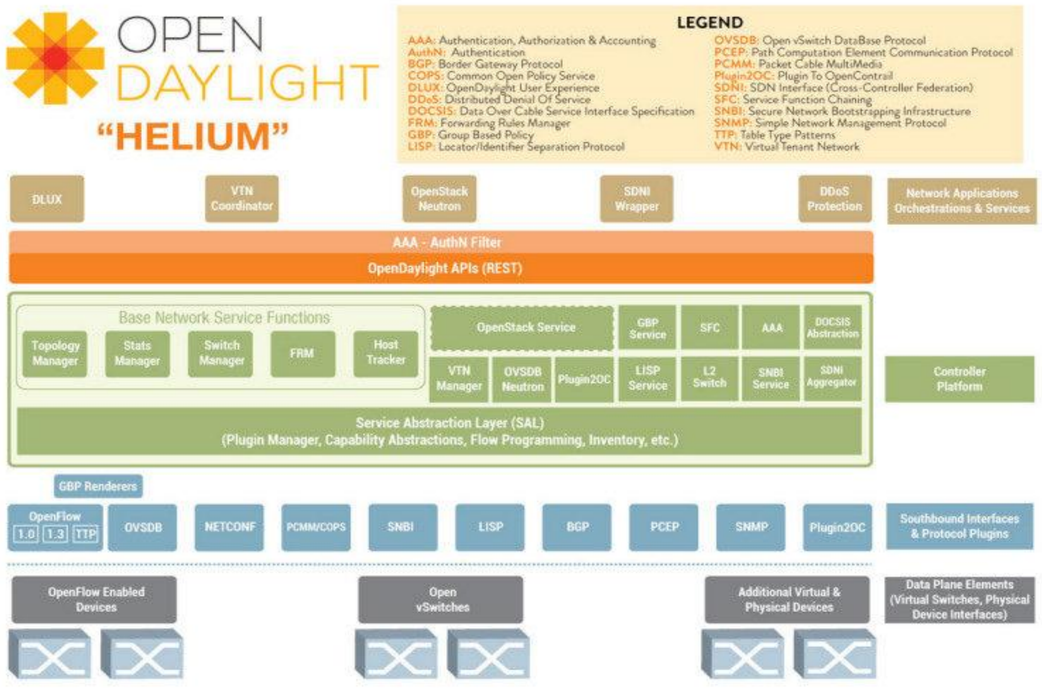
OPEN DAYLIGHT
"HELIUM"
LEGEND
AAA: Authentication, Authorization & Accounting
AuthN: Authentication
BGP: Border Gateway Protocol
COPS: Common Open Policy Service
DLUX: OpenDaylight User Experience
DDoS: Distributed Denial Of Service
DOCSIS: Data Over Cable Service Interface Specification
FRM: Forwarding Rules Manager
GBP: Group Based Policy
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch DataBase Protocol
PCEP: Path Computation Element Communication Protocol
PCMM: Packet Cable MultiMedia
Plugin2OC: Plugin To OpenContrail
SDNI: SDN Interface (Cross-Controller Federation)
SFC: Service Function Chaining
SNBI: Secure Network Bootstrapping Infrastructure
SNMP: Simple Network Management Protocol
TTP: Table Type Patterns
VTN: Virtual Tenant Network
DLUX
VTN Coordinator
OpenStack Neutron
SDNI Wrapper
DDoS Protection
Network Applications Orchestrations & Services
AAA - AuthN Filter
OpenDaylight APIs (REST)
Base Network Service Functions
Topology Manager
Stats Manager
Switch Manager
FRM
Host Tracker
OpenStack Service
GBP Service
SFC
AAA
DOCSIS Abstraction
VTN Manager
OVSDB Neutron
Plugin2OC
LISP Service
L2 Switch
SNBI Service
SDNI Aggregator
Controller Platform
Service Abstraction Layer (SAL)
(Plugin Manager, Capability Abstractions, Flow Programming, Inventory, etc.)
GBP Renderers
OpenFlow
1.0
1.3
TTP
OVSDB
NETCONF
PCMM/COPS
SNBI
LISP
BGP
PCEP
SNMP
Plugin2OC
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

Figure 2: The complete view of OpenDaylight (ODL) architecture.

3.1 Controller Platform

The controller platform is act like a middleware in the OpenDaylight ecosystem. The ODL is modular, pluggable, and flexible controller, the Controller platform is the main layer in its architecture which enables SDN abstraction. This layer displays Open Northbound (NB) APIs to the network applications to control and manage the physical and virtual elements within the network. It also consist of the Base Network Service Functions, The Platform Network Service Functions, Extensions, and the Service Abstraction Layer (SAL) [3].

A. The Base Network Service Functions

The controller platform communicates with the basic network infrastructures using southbound plugins. This service function has various basic network functions. It contains set of managers displayed in the base network service functions section. Figure 1, including Topology Manager, Switch Manager, etc. these network service functions can be used by custom application.

- **Topology Manager:** When the controller platform starts, the Topology Manager creates the root node in the topology operational subtree and subtree stores and handles information about the managed networking devices. It creates this subtree by listening to the notifications when a switches is added or removed with their interconnection status [4].
- **Statistics Manager:** This manager executes statistics collection. It sends statistics requests to all managed switches and stores all responses in the statistics operational subtree. Statistical information on switch ports, meter, tables, and flows are provided by Statistics Manager [3].
- **Switch manager:** This manager provides switches and switch ports details. When controller finds network components, their parameters are stores to the Switch Manager data tree. We can use northbound APIs to get information on the located switches and port devices [3].
- **Forwarding Rules Manager (FRM):** This network services manages basic updated rules (OpenFlow rules), resolves their conflicts, and validate them.
- **Host Tracker:** This network service stores detail data about end hosts. (Data layer address, switch type, port type, network address), and provide APIs that fetch end switches information. Host tracker uses mac address to track the status of the database. Or populated manually via NB APIs [4].

B. The Platform Network Service Functions

ODL controller contains pluggable oriented services which perform specific networking tasks and other extensions to enhance the SDN functionality. In a platform services following some services are there.

- **Affinity Metadata Service:** provides a NB APIto express network requirements of applications and communicating workload to the controller [5]**.**
- **Virtual Telnet Network (VTN) Manager:** it is an application that provides a multi-tenant virtual network on an SDN controller. It allows users to define a network with a look and feel of the conventional L2/L3 network.
- **LISP Flow Mapping Service:** it provides LISP mapping system service. This includes LISP map-server and LISP Map-Resolver services to OpenDayLight applications. Mapping data can also include a variety of routing policies including traffic engineering and load balancing [5].
- **OVSDB Neutron:** The ODL OVSDB manager interacts with the ovsdb-server and theODLOpenFlow controller interacts with the ovs-vswitchd process. The OVSDB southbound plugin plugs into the ovsdb-server. All the configuration of Open vSwitch is done with OVSDB and all the flow adding/removing is carried out with OpenFlow [6].
- **OpenStack Service:** OpenStack Networking (neutron) supports a plugin model that allows it to integrate with multiple different systems in order to implement networking capabilities for OpenStack. ODL display a single common NB service, which is implemented by neutron NB component. The exposed API matches exactly the REST API of neutron [7].
- **Authentication, Authorization, and Accounting (AAA) Service:** The AAA framework compares a user's authentication credentials stored in a database. If the credentials match, the user is allowed access to the network if the credentials don't match, authentication fails and access is denied. Authorization is the process of finding out what an authenticated user is allowed to do within the system, which tasks can do, which API can call, etc. The authorization process determines whether the user has the authority to perform such actions. Accounting is the process of recording the activity of an authenticated user [8].

C. The Service Abstraction Layer

The central concept of the OpenDaylight controller is the Service Abstraction Layer (SAL), which connects the protocol plugins and Service Network Function Modules. Because the original API-Driven SAL (AD-SAL) approach proved ineffective, OpenDaylight and all dependent projects are migrating to Model-Driven SAL (MD-SAL), which is now our focus.

Application-Driven SAL (AD-SAL)

The developers of ODL started coding the original SAL with an API-Driven SAL architecture [8]. The developers had to code the SAL APIs (to route service requests between consumers and providers) and the adaptation functionality (if NB (Service, abstract) API is not similar to its corresponding SB (protocol) API) [8]. Despite AD-SAL hiding the element-level complexity, the scalability of ODL can be limited by coding the SAL APIs as well as the adaptation functionality of each new plugin each time.

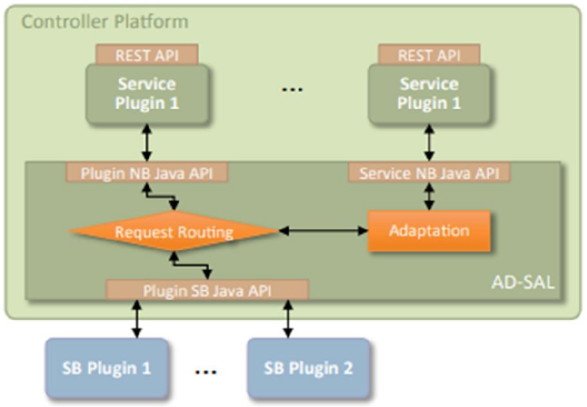

Figure 3: API-Driven SAL (AD-SAL)

Model-Driven SAL (MD-SAL)

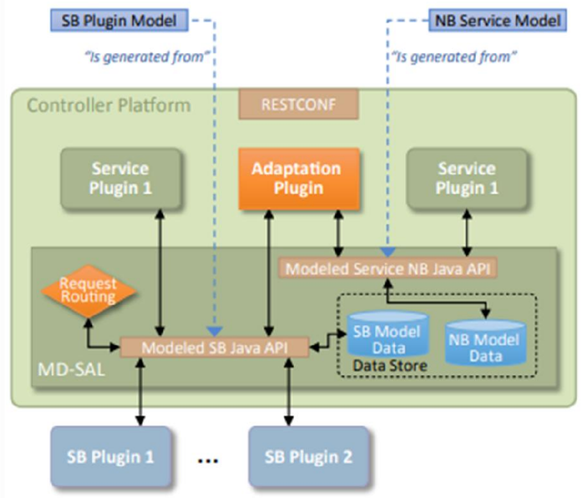

Figure 4: Model-Driven SAL (MD-SAL)

As a result of the demand for a scalable architecture, a new model-driven architecture of SAL is implemented (referred to as Model-Driven SAL or MD-SAL) [8]. The MD-SAL provides common approach to plugin development, enabling unification between both NB and SB APIs and the data structures used in various components of the controller. In MD-SAL, all status-related details stored in the form of a document object model (DOM), known as a "data tree". MD-SAL uses YANG, a single schema language, as a modelling language for describing all network services [4].

3.2 The Southbound Interface and Protocols Plugins

The southbound interfaces is capable of supporting multiple protocols (as separate plugins) these modules are dynamically linked to a service abstraction layer (SAL), which determines how to fulfill the service requested (by applications) irrespective of the basic protocol used between the controller and the network devices.

- **OpenFlow Plugin:** It includes (a) connection, session, and state managers to manage the connection with the switches, (b) an error handing mechanism, (c) a packet handler to handle incoming packets from the switches, and (d) a set of basic services such as flow, stats, and topology. This OpenFlow plugin, along with the OpenFlow protocol libraries, will constitute the southbound OF 1.0/1.3 module [3].
- **SNMP Plugin:** Single Network Management Protocol (SNMP) uses Net-SNMP library to read values from. Network devices. The SNMP input plugin supports SNMP v1, v2c, and v3 over UDP and TCP transport protocols.
- **BGP-LS Plugins:** it implements java-based Border Gateway Protocol. (BGP) and Path Computation Element Protocol (PCEP). BGP–LS plugin supports BGP Linkstate Distribution and considered for The ODL as a source of L3 topology information.

Network Configuration Protocol (NETCONF) Plugin

ODL provides a southbound Network Configuration Protocol (NETCONF) connecter API, which uses NETCONF and YANG models, to interact with a network devices. And also it used to interact with SB devices of ODL controller [9].

3.3 The Network Application and Services

The top layer of OpenDaylight consists of business and network logic applications that control and monitor network behavior. Most of the applications are mapped to the corresponding platform services, such as a VTN coordinator and VTN manager, an SDNI wrapper and aggregator, etc. In addition, the NB applications include more complex orchestration applications that engineer network traffic in accordance with the needs of environment such as the cloud and NFV [3].

- **OpenDaylight User experience (DLUX):** DLUX is an openflow network management application for Opendaylight controller. It is web-based user interface (UI) for the ODL's second release "Helium". DLUX also uses the SAL services to obtain network related information and use it to provide network management capabilities [10].
- **VTN Coordinator**: The VTN coordinator is an external application that provides a REST interface for user to use OpenDaylight VTN virtualization. It interacts with VTN Manager Plugin to implement the user configuration.
- **SDNi Wrapper**: SDNi BGP Wrapper will be responsible for the sharing and collecting information to/from merge controllers. This wrapper utilizes the existing ODL-BGP plugin. NB SDNi plugin acts as an aggregator for collecting network information such as topology, stats, host etc. this data to be exchanged is available through the RestAPIs that are developed. Wrapper reads and stores this data in a database (SQLite) [10].
- **DDoS Protection:** Defence4all is an SDN application for detecting and mitigating Distributed Denial of Services (DDoS) attacks. The application communicates with ODL controller through the northbound REST API [1].

IV. Conclusion

In the above article, we presented a summary on OpenDaylight controller with the SDN and NFV. In the beginning we discussed overview of ODL. OpenDaylight Helium components include a fully pluggable controller, interfaces, protocol plug-ins, and applications. The Helium controller has three blocks the ODL controller platform, Northbound applications and services, Southbound plugins and protocols. After that we discussed SDN working of SDN and its architecture Evolution fromSD-SAL to MD-SAL plugin in SAL framework we discussed. Each and every components of the OpenDaylight Architecture in a simplified way has covered in this paper.

References

[1]. https://nexus.opendaylight.org/content/sites/site/org.opendaylight.docs/master/userguide/manuals/userguide/bk-user-guide/content/_opendaylight_controller_overview.html
[2]. https://www.ciena.com/insights/what-is/What-Is-SDN.

[3]. https://thenewstack.io/sdn-series-part-vi-opendaylight/
[4]. https://www.mirantis.com/blog/whats-opendaylight/
[5]. https://docs.opendaylight.org/projects/lispflowmapping/en/latest/user-guide.html
[6]. https://web.archive.org/web/20201219224645/https://network-insight.net/2016/04/opendaylight-neutron/
[7]. https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/red_hat_opendaylight_product_guide/how_does_opendaylight_cooperate_with_openstack
[8]. https://docs.opendaylight.org/projects/aaa/en/latest/dev-guide.html#overview
[9]. https://www.juniper.net/documentation/us/en/software/junos/netconf/topics/task/configure-odl-integration.html
[10]. https://events.static.linuxfound.org/sites/events/files/slides/ODL-SDNi_0.pdf