OPEN DAYLIGHT SUMMIT

OPEN DAYLIGHT SUMMIT

Inter SDN Controller Communication (SDNi)

Rafat Jahan, R&D Lead(SDN), Tata Consultancy Services

#ODSummit

Introduction

Need for SDNi

- SDN will be deployed in large-scale networks, where it is divided into multiple connected SDN domains, for better scalability and security.
- To utilize network resources efficiently, SDN controllers need to communicate with each other.
- Networks are moving towards cloud architecture, where multiple SDN domains will be prevalent.
- Enabling this communication in ODL encourages adoption of OpenDaylight controller.

Sample Business Use Cases

Bandwidth on Demand

When the network resources are distributed among multiple SDN domains, controllers from each domain need to communicate with the controller of the source domain, to share network parameters. This enables the controller of the source domain to confirm and process the bandwidth requirement.

Content Delivery Networks

Service Providers have to meet the content delivery requirements as per the committed QoS. If the CDN server or cache nearest to the customer location is experiencing high loads and is unable to serve the customer, the request is sent to a CDN server/ cache that is not loaded. However, this CDN Server/ Cache may be located in a different SDN domain in the network. Hence, the source SDN controller needs to communicate (over SDNi) with the other SDN controllers within the network to negotiate a path to the best possible CDN server/ cache that meets the customer's QoS expectation.

Separate SDN Controllers for Various Networks

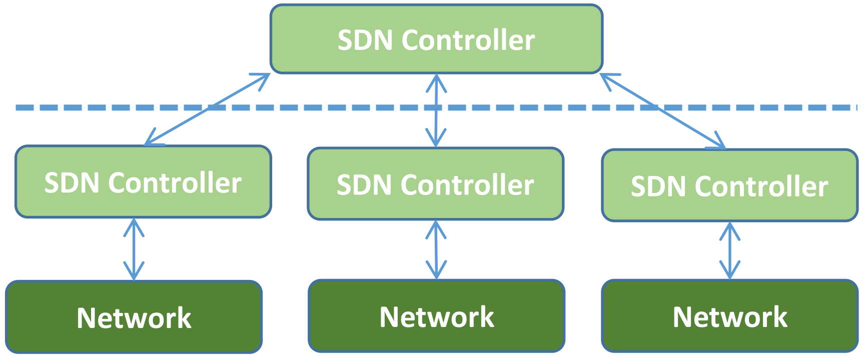
If the customer (retail, enterprise) demands a service that is provided by the data center, the SDN controllers of access, edge, core networks, and the data center need to communicate with each other. SDN controllers need to pass on parameters like QoS and policy information from the access network to the core network to the data center.

Potential Implementation Approaches

Inter-SDN controller communication is enabled by:

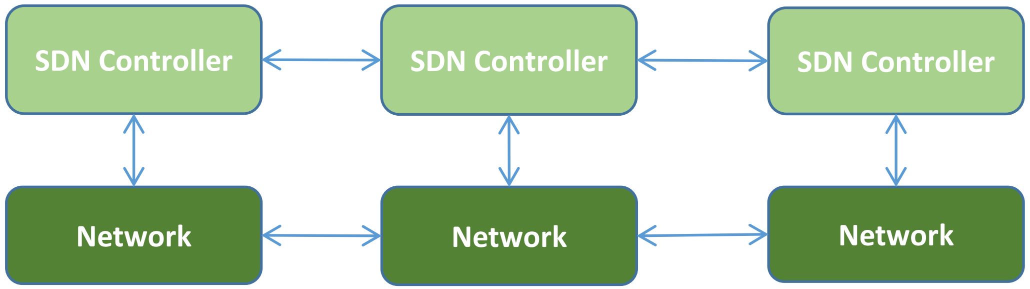
- Establishing inter-AS domain communication
- Enabling SDNi (Software Defined Network interface) for ODL, as an application (ODL-SDNi App)

Vertical Approach



- A master controller is over the individual network controllers.
- Master controller has a global view of the network across all connected SDN domains.
- It can orchestrate the configuration in each connected SDN domain.

Horizontal Approach



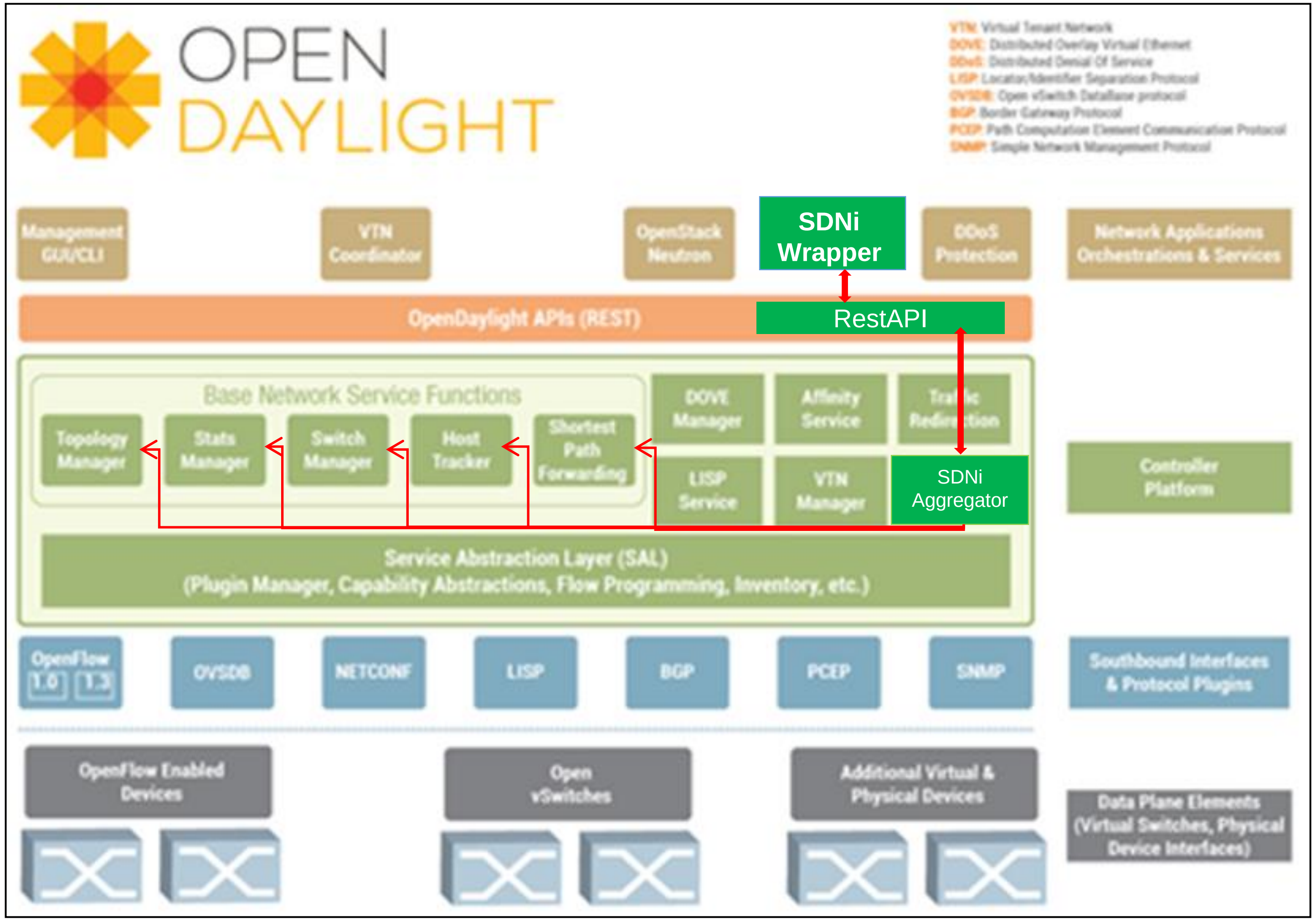
- In the horizontal approach, the SDN controllers establish peer-to-peer communication.
- Each controller can request for information or connections from SDN controllers of other domains in the network.
- This is also called the SDN east-west interface.

Architecture

OPEN DAYLIGHT SUMMIT

#ODSummit

ODL-SDNi Architecture


OPEN DAYLIGHT
VTN: Virtual Tenant Network
DOVE: Distributed Overlay Virtual Ethernet
DDoS: Distributed Denial Of Service
LISP: Locator/Identifier Separation Protocol
OVSDB: Open vSwitch DataBase protocol
BGP: Border Gateway Protocol
PCEP: Path Computation Element Communication Protocol
SNMP: Simple Network Management Protocol
Management GUI/CLI
VTN Coordinator
OpenStack Neutron
SDNi Wrapper
DDoS Protection
Network Applications Orchestrations & Services
OpenDaylight APIs (REST)
RestAPI
Base Network Service Functions
Topology Manager
Stats Manager
Switch Manager
Host Tracker
Shortest Path Forwarding
DOVE Manager
Affinity Service
Traffic Redirection
LISP Service
VTN Manager
SDNi Aggregator
Controller Platform
Service Abstraction Layer (SAL)
(Plugin Manager, Capability Abstractions, Flow Programming, Inventory, etc.)
OpenFlow 1.0 1.3
OVSDB
NETCONF
LISP
BGP
PCEP
SNMP
Southbound Interfaces & Protocol Plugins
OpenFlow Enabled Devices
Open vSwitches
Additional Virtual & Physical Devices
Data Plane Elements (Virtual Switches, Physical Device Interfaces)

SDNi Aggregator:

- Northbound SDNi plugin acts as an aggregator for collecting network information such as topology, stats, host etc.
- This plugin is extendable as per needs of network parameters required to be shared.

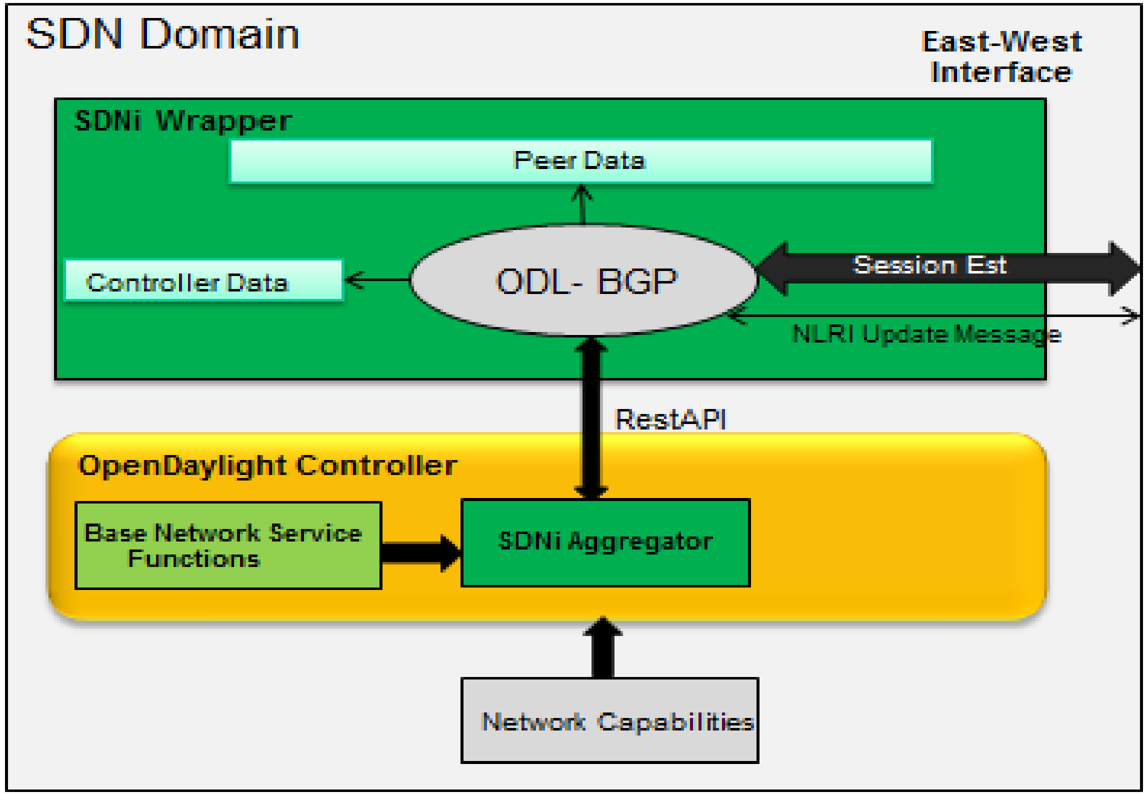
SDNi RestAPI:

- SDNi REST APIs is implemented to fetch the aggregated information from the SDNi aggregator.
- New SDNi RestAPI can be added, to support new network parameters.

SDNi Wrapper:

- SDNi BGP Wrapper will be responsible for the sharing and collecting information to/from federated controllers.

How ODL-SDNi works



Wrapper Features:

- SDNi Wrapper utilizes the existing ODL-BGP Plugin.
- Enhanced the NLRI update message (of BGP) for capability data
- This data to be exchanged available through the RestAPIs that are developed.
- Wrapper to read and store this data in a database (SQLite).
- Each controller to have peer data for the controllers in a session over real-time.
- The data exchanged can be restricted (based on security)

Network Parameters Supported

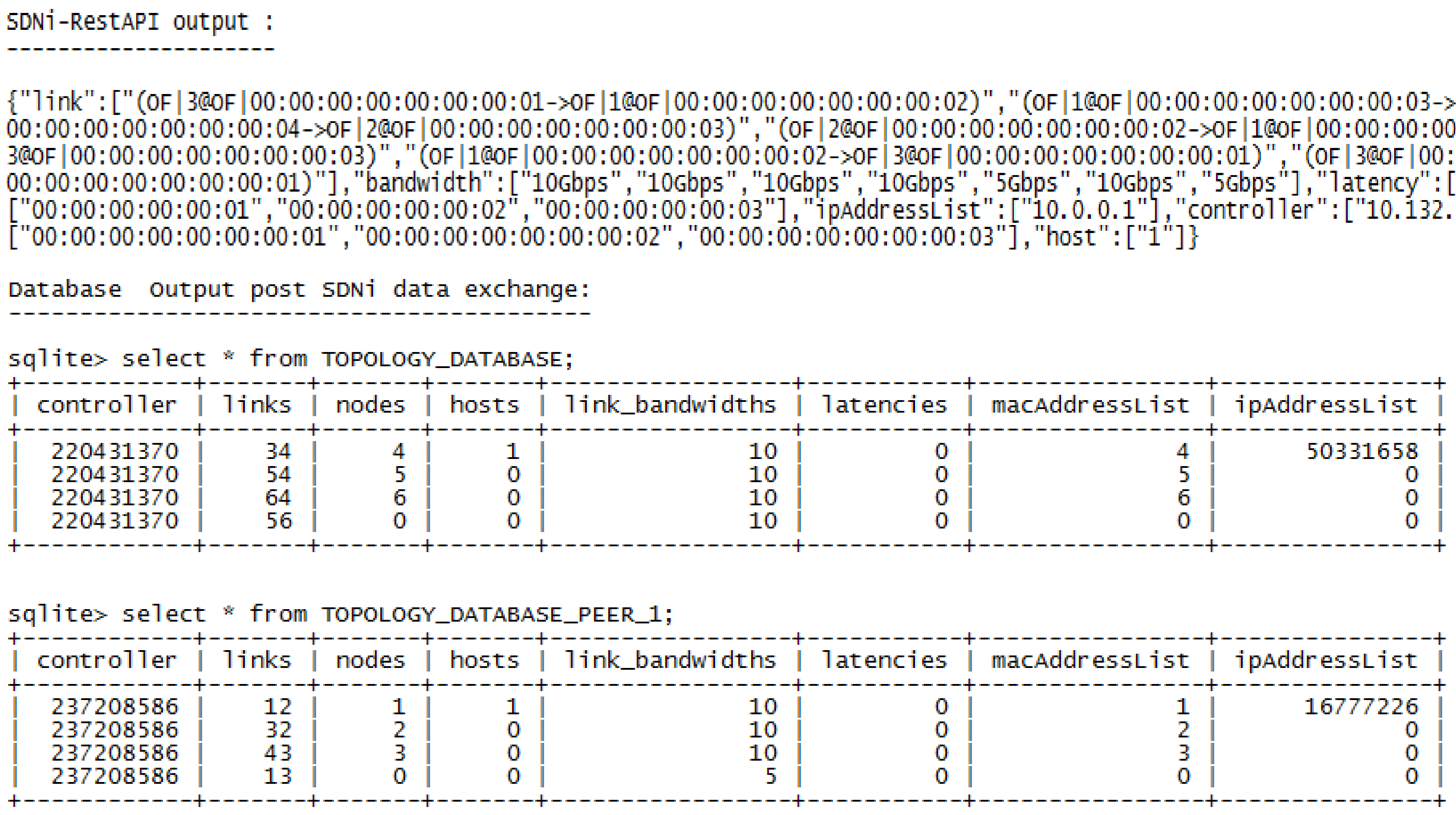
Controllers need to exchange information such as:

- Reachability update
- Flow setup, tear-down, and update requests
- Capability Update Information

Network parameters currently supported in ODL-SDNi:

Topology Data	QoS Data	Extended to (Examples)
• Controller IP Address • Links • Nodes • Link Bandwidths • MAC Address of switches • Latency • Host IP address	• Packet Loss rate • Packets Transmitted • Packets Received • Collision Count • Packets Delay	• Network topology • Network events • User defined request information • QoS requirements from user application request

Sample Output



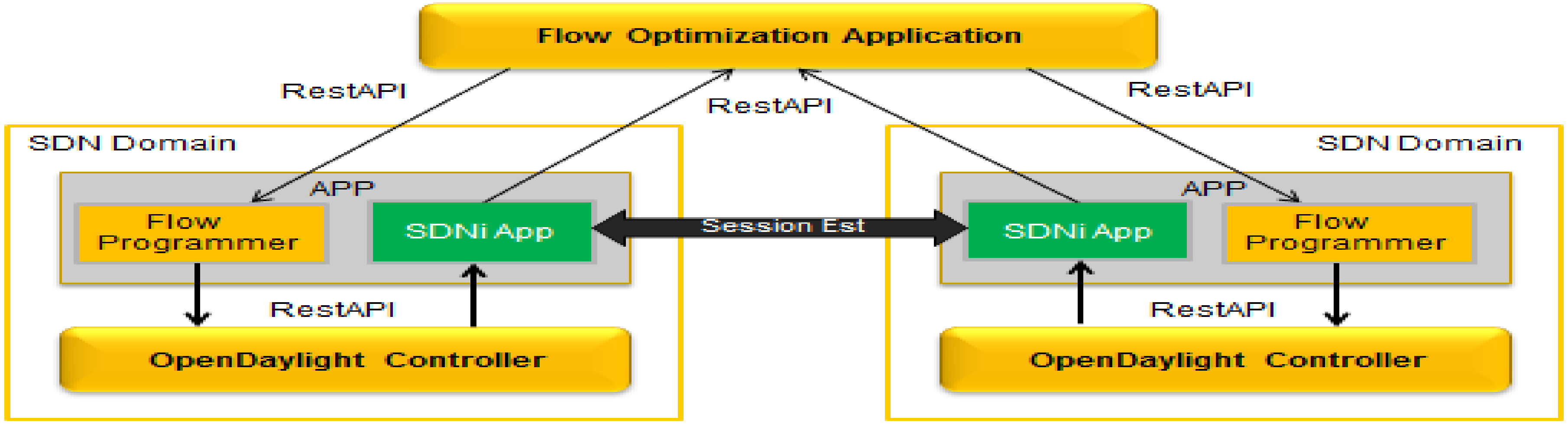
```
SDNi-RestAPI output :
---------------------

{"link":["(OF|3@OF|00:00:00:00:00:00:00:01->OF|1@OF|00:00:00:00:00:00:00:02)","(OF|1@OF|00:00:00:00:00:00:00:03->OF|2@OF|00:00:00:00:00:00:00:02)","(OF|1@OF|
00:00:00:00:00:00:00:04->OF|2@OF|00:00:00:00:00:00:00:03)","(OF|2@OF|00:00:00:00:00:00:00:02->OF|1@OF|00:00:00:00:00:00:00:03)","(OF|4@OF|00:00:00:00:00:00:00:01->OF|
3@OF|00:00:00:00:00:00:00:03)","(OF|1@OF|00:00:00:00:00:00:00:02->OF|3@OF|00:00:00:00:00:00:00:01)","(OF|3@OF|00:00:00:00:00:00:00:03->OF|4@OF|
00:00:00:00:00:00:00:01)"],"bandwidth":["10Gbps","10Gbps","10Gbps","10Gbps","5Gbps","10Gbps","5Gbps"],"latency":[],"macAddressList":
["00:00:00:00:00:01","00:00:00:00:00:02","00:00:00:00:00:03"],"ipAddressList":["10.0.0.1"],"controller":["10.132.35.14"],"node":
["00:00:00:00:00:00:00:01","00:00:00:00:00:00:00:02","00:00:00:00:00:00:00:03"],"host":["1"]}

Database  Output post SDNi data exchange:
-----------------------------------------

sqlite> select * from TOPOLOGY_DATABASE;
+------------+-------+-------+-------+-----------------+-----------+----------------+---------------+
| controller | links | nodes | hosts | link_bandwidths | latencies | macAddressList | ipAddressList |
+------------+-------+-------+-------+-----------------+-----------+----------------+---------------+
|  220431370 |    34 |     4 |     1 |              10 |         0 |              4 |      50331658 |
|  220431370 |    54 |     5 |     0 |              10 |         0 |              5 |             0 |
|  220431370 |    64 |     6 |     0 |              10 |         0 |              6 |             0 |
|  220431370 |    56 |     0 |     0 |              10 |         0 |              0 |             0 |
+------------+-------+-------+-------+-----------------+-----------+----------------+---------------+


sqlite> select * from TOPOLOGY_DATABASE_PEER_1;
+------------+-------+-------+-------+-----------------+-----------+----------------+---------------+
| controller | links | nodes | hosts | link_bandwidths | latencies | macAddressList | ipAddressList |
+------------+-------+-------+-------+-----------------+-----------+----------------+---------------+
|  237208586 |    12 |     1 |     1 |              10 |         0 |              1 |      16777226 |
|  237208586 |    32 |     2 |     0 |              10 |         0 |              2 |             0 |
|  237208586 |    43 |     3 |     0 |              10 |         0 |              3 |             0 |
|  237208586 |    13 |     0 |     0 |               5 |         0 |              0 |             0 |
+------------+-------+-------+-------+-----------------+-----------+----------------+---------------+
```

OPEN DAYLIGHT SUMMIT

#ODSummit

Demonstration

Demo



Business Challenge

- In the multi-vendor environment, network traffic needs to be orchestrated across intra-/inter-domain subnets of the SDN controllers.
- The mandate is of a east-west communication that enables SDN controllers across subnets to exchange network information within the purview of defined policies
- Inter-SDN controller (multi-vendor) communication – exchange of network parameters needs to be per pre-agree interface specifications

Release and References

ODL Releases

Helium Release:

- The ODL-SDNi Application was providing an east-west interface among multiple OpenDaylight controllers.
- The communication was established by exchange of network parameters over NLRI Update message of BGP protocol(RFC 4271) .
- Controller Topology data exchanged over SDNi

RestAPI URL: http://localhost:8080/controller/nb/v2/sdni/default/topology

Lithium Release:

- Implementation of QoS data exchange over SDNi
- Update the Aggregator and RestAPI with respect to QoS data.
- Implement database for maintaining self and peer controller information in the database for easy access from application above.

RestAPI URL: http://localhost:8080/controller/nb/v2/sdni/default/qos

Roadmap for further Releases:

- ODL-SDNi to have a user interface.
- Migration of SDNi from AD-SAL to MD-SAL
- Incorporate Traffic Steering by conveying controller tear down over SDNi Application.

References

Documents to refer for ODL-SDNi set up:

The ODL-SDNi feature can be done with Helium and Lithium releases over SP edition (no more supported) And Karaf.

- Project Proposal: https://wiki.opendaylight.org/view/Project_Proposals:ODL-SDNi_App
- User Guide: https://wiki.opendaylight.org/view/ODL-SDNiApp:User_Guide
- Developer Guide: https://wiki.opendaylight.org/view/ODL-SDNiApp:Developer_Guide

Thank You

OPEN DAYLIGHT SUMMIT

#ODSummit