SDN Ground Truth:
Implementing a Massive Scale Programmable DOCSIS Network

Sameer Patel
Manager
Network Policy Engineering – T&P
Comcast

Jason Schnitzer
Founder & CEO
Applied Broadband

Mohammad Kabir Chowdhury
Product Engineer
Comcast

Dr. David Early
Data Scientist
Applied Broadband

Abstract

SDN carries the promise of enhanced and expanded services with a more manageable operational environment. Though many standards groups and open source projects exist, deployments remain in their infancy.

This paper provides an overview of Comcast's SDN-driven implementation of programmable Service Flows through the integration of the OpenDaylight PacketCable Multimedia plugin. This paper also provides practical considerations for successful analysis of SDN architectures and technologies when applied to DOCSIS Broadband network deployments.

MOTIVATION

Making The Case for SDN and DOCSIS

The application of a software-defined network (SDN) architecture to DOCSIS [1] broadband access allows resources to be virtualized and allocated to whatever application or service requires them.

Though advantages to SDN-based architectures and applications have been proposed [2], the specific benefits when applied to a programmable DOCSIS policy environment are as follows:

1. **System and network resource elasticity**. Traditionally, deployments of network provisioning and management software for DOCSIS networks has been burdened by rigid environmental constraints and restrictive resource dimensioning limitations. The virtualization of DOCSIS low-level services using contemporary SDN architectures enables greater design flexibility in networks of scale, while affording discrete capacity upgrades at time of need.

2. **A common platform across network applications.** Capital, operational, and organizational economies of scale result when using multiple applications that access the same network services and related resources in a common way.

3. **Service delivery agility**. As a consequence of both (1) and (2) above, new subscriber services that benefit from

QoS treatment in the access network can quickly be supported.

Business agility is essential to the very survival of broadband organizations, therefore the ability to implement new services quickly is vital.

Traditional DOCSIS network architectures and services lack the flexibility required to support an agile business, and over-provisioning the network for peak utilization inflates costs. A forward-looking SDN approach is required for any organization seeking to become agile, because this approach brings dynamic features to the network and gets maximum benefit from network virtualization functions for a minimum overall cost. Virtualization has largely transformed the data center with flexible and automated server provisioning, but networking and storage infrastructure have not kept pace.

Current Use Cases

Within the cable access network, SDN has the potential to reduce service deployment and maintenance times, reduce errors associated with those setups, offer new management options, and facilitate the introduction of more fine-grained service offerings. While SDN can apply to a wide array of applications in the cable access network [3], our implementation addresses three specific use cases:

- **Telephony** – The application of QoS to enhance managed VoIP traffic.
- **Congestion management** – Dynamic, rule-based implementation policy controls to assure fair access to resources during times of network congestion.
- **Acceptable Use Policy Management (AUPM)** – The application of network-based controls to address service abuse.

APPLYING SDN PRINCIPLES TO BROADBAND NETWORKS

SDN (at least in today's leading initiatives) finds its genesis in switch networks largely aimed at enterprise network environments. OpenFlow [4] is the leading standard, driving many SDN implementations today. OpenFlow is largely concerned with the programming of packet paths (flows) across a network of IP switches.

When comparing OpenFlow to Cable access network technologies such as DOCSIS, fundamental differences exist:

- **Topology:** The underpinning HFC network is a physical tree/branch topology. The CMTS (master) and CM (slave) topology, and resulting protocol design, are fundamentally different than port-to-port topology of switched networks such as Ethernet.
- **Scale**: The number of managed devices (entities) in a single managed DOCSIS network is well into the tens of millions of and expected to grow further.
- **Complexity**: The information model representing a complete DOCSIS network describes objects and attributes numbering in the thousands.

The following section provides a brief overview of basic SDN principles. Readers familiar with these concepts are encouraged to skip to the next section.

The SDN architecture abstraction presented by the ITU-T in [5] describes three functional layers and two interfaces between the layers (illustrated in Figure 1). The layers are:

1. **Application Layer:** SDN and/or business applications that specify network behavior.

2. **SDN Control Layer:** "a means to dynamically and deterministically control the behavior of network resources…as instructed by the application layer" [5].

3. **Resource Layer:** Network devices and element management systems.

The interfaces – the Application-Control Interface and the Resource-Control Interface – represent bi-directional communication between the components of each logical layer.

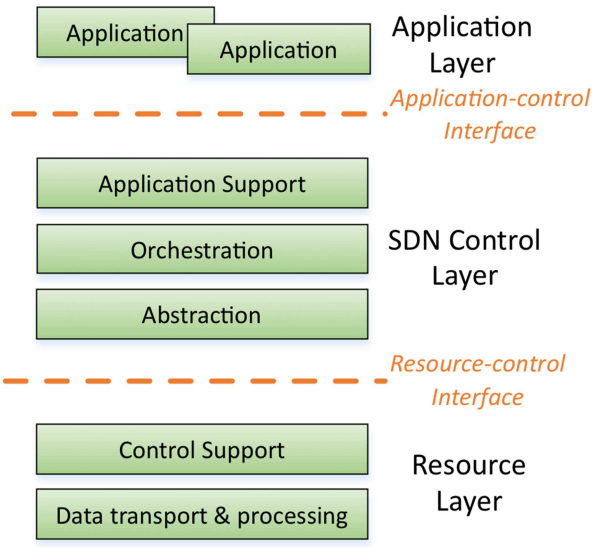

Figure 1: SDN High Level Abstraction (ITU-T Y.3300 [5])

For the purposes of our implementation, the architecture is consistent with the general description above with the following additional details:

1. The Application Layer includes a total of three (3) software applications, each representing a current use case.

2. Each of the three Applications communicate with the SDN Control Layer by way of an API based on a common data model abstracting the Resource Layer (network).

3. The SDN Control Layer forwards Application requests with minimal modifications to the incoming messages. The Control Layer is also responsible for the storage and management of both the desired Resource Layer configuration and the actual operational state.

4. The Resource Layer is implemented by the DOCSIS network infrastructure which features control support as well as data transport and processing.

ARCHITECTURE

This new programmable architecture proposes enhancements to current DOCSIS access network Policy Management infrastructure in order to support both legacy and emerging use cases by leveraging Software Defined Networking (SDN) architectural principles.

The system architecture described in this section presents an overall vision for how the OpenDaylight (ODL) [6] SDN platform is used to implement specific PCMM functions within the DOCSIS policy infrastructure. The proposed architecture is aligned with next-generation access systems and programmable network goals. The architecture leverages new development and operations models to create a more durable, open, and extensible policy infrastructure for DOCSIS deployments.

OpenDaylight

The OpenDaylight (ODL) [6] initiative is a collaborative open source project that aims to accelerate adoption of SDN and create a solid foundation for Network Functions Virtualization (NFV) enabling a more transparent approach that fosters new innovation and reduces risk. Founded by industry leaders and open to all, the

OpenDaylight community is developing a common, open SDN framework consisting of code and blueprints.

Porting PCMM to ODL

Though a rigorous review of the CableLabs PacketCable Multimedia (PCMM) [7] protocol will not be provided in this paper, it is useful to understand a few key concepts in the context of integrating DOCSIS and PCMM capabilities into an SDN platform architecture. Originally released in 2003, the intent of the PCMM specification was to support the deployment of general Multimedia services by providing a technical definition of several IP-based signaling interfaces that leverage core QoS and policy management capabilities native to DOCSIS Versions 1.1 and greater.

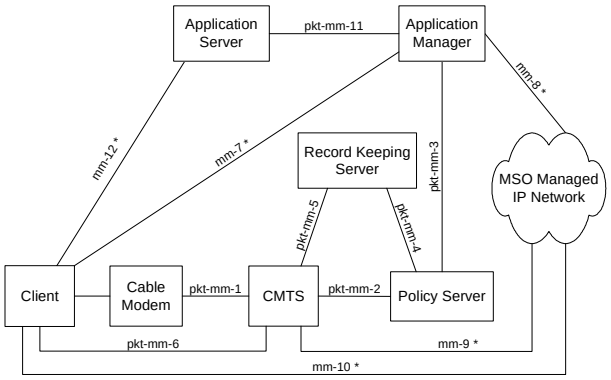

Figure 2: Complete PCMM Architectural Framework

Figure 2 illustrates the complete PCMM architectural framework, defining a collection of elements and interfaces which together support the service objectives of Dynamic QoS (DQoS)-enabled capabilities in DOCSIS access networks.

A limited subset of the overall capability described in the most current PCMM release [7] is needed to support the use cases and requirements necessary for SDN implementation. For the purposes of our application, only three (3) of the seven (7) elements formally described in the PCMM architectural framework are addressed:

1. Cable Modem (CM),

2. Cable Modem Termination System (CMTS), and

3. Policy Server (PS).

It follows that a small subset of the interfaces (two of the twelve total) are relevant to future SDN applications:

1. **pkt-mm-1:** Using CMTS to CM DOCSIS messaging, the CMTS instructs the CM to setup, teardown or change a DOCSIS service flow in order to satisfy a QoS request via DSx signaling.

2. **pkt-mm-2:** This interface (PS to CMTS) controls policy decisions, which may be: (a) pushed by the Policy Server (PS) onto the CMTS, or (b) pulled from the PS by the CMTS. In some scenarios, this interface may also be used to inform the PS when QoS resources have become inactive. Common Open Policy Service (COPS) [8][9][10] is used to transfer policy information between the PS and the CMTS. The PS initiates communication for a Multimedia session by sending a DQoS Gate-Set message (which is an unsolicited COPS message) to the CMTS.

It is worth noting that, along with the unnecessary interfaces and elements, other historically relevant concepts used to describe PCMM capabilities were not carried forward into the SDN-based architecture. These include more rigid COPS/PCMM-centric concepts such as Service Control Domain (SCD), Resource Control Domain (RCD), Policy Decision Point (PDP) and Policy Enforcement Point (PEP). In some cases, these protocol semantics were preserved but

are now expressed using different structures and terminologies within a new SDN context.

OpenDaylight PCMM Plugin

Figure 3 represents a high-level view of selected PCMM capabilities within the context of the ODL's SDN Controller architecture. In it, the PCMM capabilities to be integrated into the ODL Controller take the form of a PCMM device communication plugin as a SouthBound (SB) device communication plugin – that is, a low level protocol implementation above the network resource layer that implements the desired PCMM PS capabilities and select protocol features of the PCMM pkt-mm-2 interface. PCMM-driven applications (representing the use cases described) interact with the PCMM-plugin using the northbound (NB) abstraction layer/API as described by the PCMM YANG data model (packetcable-policy-model).

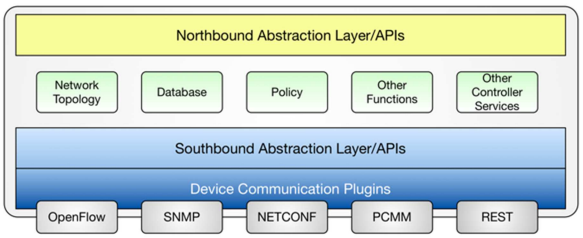

Figure 3: PCMM plug-in integrated into the ODL SDN Controller

The PCMM plugin for ODL sponsored by CableLabs was originally conceived to use the ODL platform as a proof-of-concept for the Application Manager (AM) and parts of the Policy Server, both components of the PCMM architecture. In 2014, the source code representing this reference implementation capability moved from CableLabs and became a formal PacketCable PCMM project under ODL stewardship [11].

Module	Description	Associated Feature(s) or purpose
features-packetcable-policy	Defines the bundles that are part of the Policy Servicer.	The Feature module
packetcable-driver	Contains the logic required for communicating or creating a PCMM CCAP/CMTS.	features-packetcable & features-packetcable-policy
packetcable-policy-model	Contains the YANG data model.	features-packetcable-policy
packetcable-policy-server	Responsible for processing RESTCONF calls and maintaining persistent socket connection to the CCAP/CMTS.	features-packetcable-policy
packetcable-policy-karaf	Creates the OSGi distribution for feature including ODL and other dependencies.	features-packetcable-policy
packetcable-emulator	Standard Java library that acts as an emulator for a PCMM CCAP/CMTS.	test tool

Figure 4: OpenDaylight PacketCable Multimedia (PCMM) Plugin Components

Figure 4 provides an inventory of the ODL PCMM plugin components, which together provide the following functions [12]:

- RESTCONF APIs for provisioning CMTS network elements.
- RESTCONF APIs for dynamic creation of service flows for registered CMs.
- RESTCONF APIs for dynamically controlling service flow QoS parameters.

- A complete PCMM/COP-PR messaging library and protocol stack implementation

IMPLEMENTATION

In this section we will provide an example of a complete PCMM QoS session created and terminated using the ODL system described. Figure 5 illustrates the example configuration which includes a DOCSIS CMTS, CM, ODL Controller, and a PCMM ODL Application.

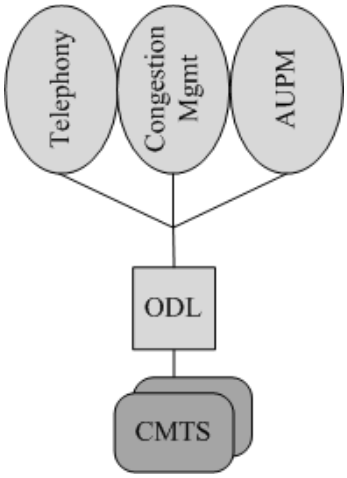

Figure 5: Connecting PCMM Applications to ODL Controller

In this example, the ODL Application is a simple HTTP client capable of performing CRUD (Create, Read, Update, Delete) operations on the ODL's internal MD-SAL datastore using data defined by the PCMM YANG module. The ODL Controller is based on the Beryllium release and includes the PCMM plugin [11][12].

The PCMM plugin extends the capabilities supported in Beryllium to implement additional PCMM protocol features not available in the open source release. Specifically, the PCMM plugin fully supports the expression of traffic profiles via a FlowSpec. A FlowSpec object defines the traffic profile associated with a Gate through an RSVP-like parameterization scheme [12].

PCMM Gate-Set

Referring to Figure 6, an Application generates a RESTCONF PUT operation to the ODL Controller's RESTCONF API exposing the YANG data model representation of the PCMM Gate-Set request (Figure 7). The content of the PUT is a JSON formatted list of parameters that together describe the desired Gate configuration (Figure 8).

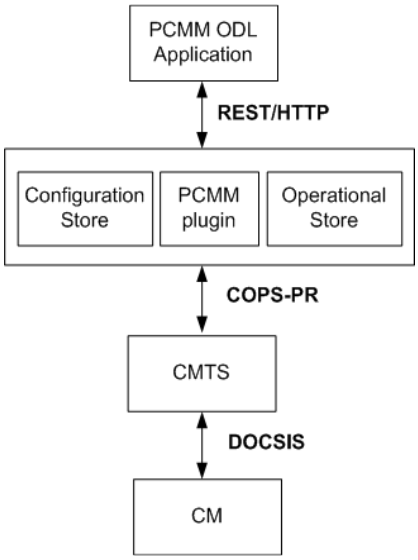

Figure 6: Example ODL PCMM System Configuration

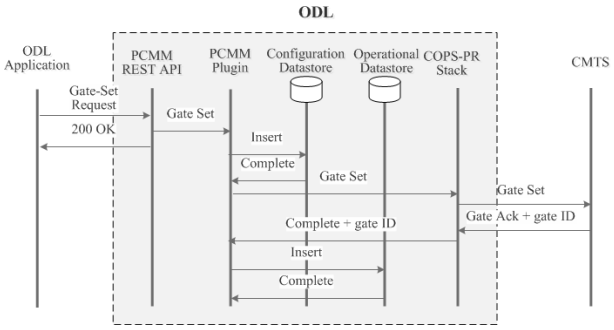

Figure 7: PCMM Gate-Set Transaction Sequence Diagram

```
http:/restconf/config/packetcable:qos/apps/app/<appId>/subscribers/subscriber/<subscriberId>/gates/gate/<gateId>/
```

Figure 8: Format of RESTCONF Gate-Set request in a HTTP PUT operation

The HTTP PUT format for an Upstream (US) FlowSpec Gate-Set request from the southbound PCMM Plugin (within ODL) to the CMTS is as follows:

- <appId> - A string used to uniquely identify the requesting application.
- <subscriberId> - The subscriber identity, usually a CM IP address.
- <gateId> - ODL Gate identifier. An internal-only reference to a unique node containing configuration parameters within the ODL datastore. The ODL gateId value is determined by the ODL Application generating the request. Note that the ODL gateId and PCMM GateId (described later in this section) are two logically isolated concepts.

Figure 9 shows example JSON content for the CM Gate-Set request. As described earlier, our use cases required a FlowSpec traffic profile, which necessitated an extension of the Beryllium release of the PCMM plugin, which only supports a Service Class Name (SCN) defined Gate profile.

The PCMM plugin NB REST API receives the Gate-Set request message from the ODL Application, and performs the following steps:

1. Basic verification of the format of the request content is made against the PCMM YANG data model. If one or more mandatory elements are missing or malformed, the request is rejected and an error is returned to the calling ODL Application.
2. Upon acceptance of the Gate-Set request message by the NB REST API, an acknowledgement message ("200 OK") is returned to the calling ODL Application. It is conceptually significant to recognize that the receipt of this acknowledgment only indicates that the REST API successfully received the message. It is not an acknowledgement that a Gate was successfully created on the CMTS.

```
{
  "gate": {
    "gateId": "<gateId>",
    "classifiers": {
      "classifier-container": [{
            "classifier-id": "1",
            "classifier": {
              "srcIp": "10.20.0.3",
              "dstIp": "10.20.0.5",
              "protocol": "0",
              "srcPort": "54322",
              "dstPort": "4322",
              "tos-byte": "0x01",
              "tos-mask": "0x00"}
        }]
    },
    "gate-spec": {
      "dscp-tos-overwrite": "0x00",
      "dscp-tos-mask": "0x00",
      "direction": "us"
    },
    "traffic-profile": {
      "flow-spec-profile": {
        "token-bucket-rate": "400",
        "token-bucket-size": "40",
        "peak-data-rate": "400000",
        "minimum-policed-unit": "400",
        "maximum-packet-size": "400",
        "rate": "30000",
        "slack-term": "0"
      }
    }
  }
}
```

Figure 9: Example JSON representation of a PCMM Gate-Set Request message

ODL Internal Message Processing

There are several steps that a request goes through within ODL, including creation and management in the internal configuration and operational datastores as defined in NETCONF [13]:

1. The NB REST API unmarshalls (deserializes) the inbound Gate-Set request.

2. ODL then semantically evaluates the contents of each parameter in the JSON content of the Gate-Set request. (Figure 9) against the constraints of the PCMM YANG model.

3. Upon successful validation, the MD-SAL creates a node within the configuration datastore and populates it with the desired Gate-Set parameters referenced by a unique ODL GateId.

4. The SB PCMM API (containing the PCMM messaging library), acting as a COPS PS or Policy Decision Point (PDP), then forms a PCMM Gate-Set message for transmission to the CMTS.

5. The CMTS receives the Gate-Set message from the PCMM Plugin, and processes accordingly.

6. The CMTS either accepts and establishes the Gate or rejects it. Upon acceptance, an acknowledgement containing a unique PCMM Gate ID is generated by the CMTS.

7. If a PCMM Gate ID is returned by the CMTS, the established Gate information, including the Gate ID, is stored in ODL's MD-SAL Operational Datastore. If no Gate ID is returned, then relevant error information is stored in the Operational Datastore.

CMTS Message Processing

As described in PCMM, the CMTS assumes the role of Policy Enforcement Point (PEP) and is responsible for processing the Gate-Set request from the ODL's SB PCMM plugin (pkt-mm-02). The CMTS executes the relevant procedure as described in [PCMM]. Once a DSA-RSP is received by the CMTS from the CM confirming a successful transaction, the CMTS will send a Gate-Set-Ack to the Policy Server. The CMTS issues a Gate-Set-Ack to the ODL PCMM plugin (PS) in response to the Gate-Set message.

If successful, the CMTS and CM collectively execute a DOCSIS DQoS Dynamic Service Add (DSA) operation to admit the Upstream Service Flow. If CMTS admission control succeeds, the CMTS will initiate the process of reserving and committing the access network resources by issuing a DSA to the Cable Modem. At this point, the requested US PCMM Gate is now installed and the traffic for the PCMM session is forwarded from the CM to the CMTS according to the DOCSIS US QoS parameters provided.

PCMM Gate-Delete

In this implementation, the ODL Application (representing one of the use cases described) is responsible for signaling in order to explicitly terminate the PCMM session.

Again assuming the role of a PCMM PS, the ODL Application terminates the session by sending a Gate-Delete message to the CMTS.

Referring to Figure 10 and Figure 11, The ODL Application generates a RESTCONF DELETE operation to the ODL Controller's RESTCONF API .

If successful, the CMTS will tear down the access network resources by sending a DSD-

REQ to the CM. The CMTS will complete the Gate-Control transaction with a Gate-Delete-Ack.

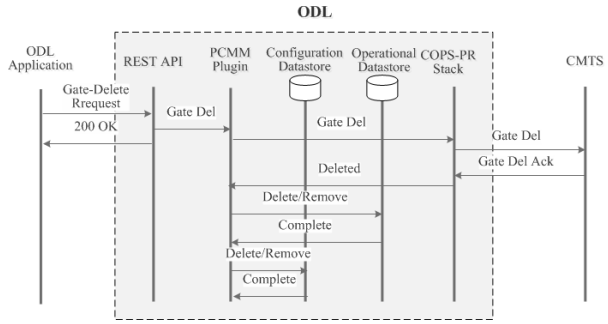

Figure 10: PCMM Gate-Delete Transaction Sequence Diagram

http:/restconf/config/packetcable:qos/apps/app/<appId>/subscribers/subscriber/<subscriberId>/gates/gate/<gateId>/

Figure 11: Format of restconf Gate-Delete request in HTTP DELETE operation

STATE OF THE ART & KNOWN LIMITATIONS

Based on the scope of the initial CableLabs PCMM plugin development, the implementation has the following current known limitations:

- Though the PCMM plugin messaging library includes full support for COPS-PR, not all PCMM Traffic Profiles and DOCSIS MAC layer scheduling types are supported in the plugin itself.
- Event messaging as described in PCMM [7] is not supported.
- Testing has been limited to small scale proof-of-concept testing. Large scale and broad functional testing is required.
- Security has not been formally addressed.
- The ODL REST API does not return any explicit (ACK/NACK) operational information to the requesting ODL Application. The only message is a "200 OK" message indicating correct syntax and acceptance of the request. Any additional information about the Gate, including whether it was successfully set, must be explicitly requested by the calling application in a separate transaction.

FUTURE WORK

Beyond addressing known limitations described in the prior section, the following areas must be addressed in order to deploy the PCMM ODL platform within a network of massive scale:

- Scalability: The role and persistence of state within the ODL Controller vs. the ODL Application layer.
- Survivability: ODL persistence and HA/DR implications.
- Manageability: Visibility and control of the system from an operations perspective in terms of fault and performance management.

Emerging Use Cases

It is anticipated that new applications providing QoS-driven enhancements to services delivered within the DOCSIS infrastructure, such as video delivery, will evolve. Each new service will be added to a common SDN infrastructure, represented by a new PCMM-based ODL Application.

Emerging Networks

DOCSIS 3.1 [13] describes a new framework for Hierarchical QoS (HQoS), which enables operators to define QoS policies on an aggregation of Service Flows. If adopted by CMTS vendors, this new mechanism for the organization of access network resources will enable new service configuration and bandwidth management scenarios not available today.

Emerging SDN Platforms

Though considerable progress has been made to realize SDN-based principles in deployment, implementations such as ODL remain in their infancy. As the realities of deploying SDN architectures into massive scale networks are learned, it is expected that new general SDN controller implementations will emerge that may address the known limitations of ODL.

CONCLUSION

In DOCSIS broadband networks, the practical application of SDN differs from those defined for other networks. The nature of the DOCSIS protocol, deployment topology, and management capabilities form the basis for unique requirements and modifications to existing SDN approaches.

To date, the integration of PCMM and DOCSIS capabilities into an SDN platform has demonstrated encouraging progress. The implementation described in this paper

highlights the functional goals of ODL acting as a control plane for the creation of DOCSIS/PCMM service flow management.

By leveraging an emerging SDN platform (ODL) and taking a modular crawl-walk-run approach to development and deployment, a practical, scalable implementation is becoming reality. While further development and field experience are required, there is a clear path forward for SDN in the new programmable DOCSIS network.

ACRONYMNS AND ABBREVIATIONS

ACK – ACKnowledgement
AM – Application Manager
API – Application Programming Interface
AUPM – Acceptable Use Policy Management
CRUD – Create, Read, Update, Delete
CM – Cable Modem
CMTS – Cable Modem Termination System
CCAP – Converged Cable Access Platform
COPS – Common Object Policy Service
COPS-PR – COPs
DOCSIS – Data Over Cable Interface Specification
DQoS – Dynamic Quality of Service
DSA – Dynamic Service Add
DSA-RSP – Dynamic Service Add Response
DSD – Dynamic Service Delete
HA – High Availability
HFC - Hybrid Fiber/Coax
HQoS – Hierarchical QoS
IETF – Internet Engineering Task Force
ITU-T – International Telecommunication Union
JSON – JavaScript Object Notation
MD-SAL – Model Driven SAL
NACK – Negative ACKnowledgement
NB – Northbound
NFV – Network Function Virtualization
ODL – OpenDaylight
PDP – Policy Decision Point
PEP – Policy Enforcement Point
PS - Policy Server
QoS – Quality of Service
RCD – Resource Control Domain
REQ – REQuest
RFC – Request for Comments
RSVP – Resource Reservation Protocol
SAL – Service Abstraction Layer
SB – Southbound
SCD – Service Control Domain
SDN – Software Defined Networking
SD – Software Defined
SF – Service Flow
VoIP – Voice Over IP

REFERENCES

[1] DOCSIS® 3.0 specifications: Cable Television Laboratories, Inc. “CableLabs Specifications” [Online] Available: http://cablelabs.com

[2] N. Feamster, J. Rexford, E. Zegura. The Road to SDN: An Intellectual History of Programmable Networks. IEEE ACM Queue, December 2013.

[3] Virtualization and Network Evolution Open Networking - SDN Architecture for Cable Access Networks Technical Report VNE-TR-SDN-ARCH-V01-150625, CableLabs.

[4] OpenFlow® [Online] Available from: http://openflow.com.

[5] Recommendation ITU-T Y.3300 (06/2014) Framework of software-defined networking, ITU, 2014.

[6] Open Daylight Project. http://www.opendaylight.org/.

[7] PacketCable Specification Multimedia Specification PKT-SP-MM-I06-110629, CableLabs, June 2011.

[8] IETF RFC 3084, COPS Usage for Policy Provisioning (COPS-PR), March 2001.

[9] IETF RFC 2748, The COPS (Common Open Policy Service) Protocol, January 2000.

[10] IETF RFC 2753, A Framework for Policy-based Admission Control, January 2000.

[11] ODL PacketCable Plugin Documentation, https://wiki.opendaylight.org/view/PacketCablePCMM:Main

[12] OpenDaylight Users Guide, https://www.opendaylight.org/sites/opendaylight/files/bk-user-guide.pdf

[13] IETF RFC6241, Network Configuration Protocol (NETCONF), June 2011.

[14] DOCSIS 3.1 MAC and Upper Layer Protocols Interface Specification, CM-SP-MULPIv3.1-I02-140320, CableLabs, 2014.