SCTE CABLE-TEC EXPO 22
SEPTEMBER 19-22 • PHILADELPHIA, PA

2022 Fall Technical Forum
SCTE • CABLELABS • NCTA

Converged Service Orchestration – Dynamic Cable Speed Boost

A Technical Paper prepared for SCTE by

Rahil Gandotra
Ph.D., Sr. Software Architect
CableLabs
858 Coal Creek Circle, Louisville, CO - 80027
(303) 661-3439
r.gandotra@cablelabs.com

Shafi Khan
Lead Software Engineer
CableLabs
858 Coal Creek Circle, Louisville, CO - 80027
(303) 661-3318
s.khan@cablelabs.com

Yunjung Yi
Ph.D., Principal Architect & Director of Wireless Standardization
CableLabs
858 Coal Creek Circle, Louisville, CO - 80027
(303) 661-3849
y.yi@cablelabs.com

Table of Contents

List of Figures

1. Introduction

Traditionally, networks serving different purposes operate in siloes with little-to-negligible overlap in their infrastructures or management processes. For example, DOCSIS networks, optical networks, and mobile networks, each employs specialized hardware functions managed by proprietary element management systems. Operators who offer both wireline and wireless services have dedicated teams to provision and manage each of their different services. This disjointed model of network management is challenged by operational economics. Consequently, designing, deploying, and operating end-to-end services are long and manual processes with long lead times (weeks to months) for effective service delivery.

On the contrary, the networks of tomorrow are envisioned to operate multiple different physical and cloud-native functions over a single flexible, programmable convergence platform whose hardware, software, and data storage resources are shared across multiple access technologies. Convergence is the process of unifying heterogeneous technologies to deliver seamless and ubiquitous connectivity. For operators, convergence creates opportunities for delivering novel, differentiated services, as well as allows for enhanced operational agility.

The umbrella term convergence consists of multiple building blocks such as access convergence, transport convergence, core convergence, platform convergence, operations convergence, and security convergence. Operations convergence entails creating a common operations framework for deploying, configuring, and managing network functions constituting a service. To make business models associated with converged networks and services increasingly attractive, in both wired and wireless domains, strategic research is required to devise the best integration architectures and appropriate accompanying integrated operations and management solutions.

When it comes to solving these challenges, technologies like software-defined networking (SDN) and network functions virtualization (NFV) have already addressed certain pieces of the puzzle. SDN enables decoupling the control plane (signaling/routing traffic) from the user plane (data/application traffic) to provide a global view of the network for efficient centralized control and allows for simpler forwarding devices. NFV allows for the separation of network functions from the underlying hardware and enables running them as virtual machines (VMs) or containers on commercial off-the-shelf (COTS) hardware. Additionally, the cloud computing paradigm provides efficient means to offer flexible resources and economies of scale. A converged service operator would need the capability to integrate all these disparate technologies into a single comprehensive framework to model end-to-end services and to abstract and automate the control and management of physical and virtual resources.

The Converged Service Management Layer (CSML) project — a key piece of the operations convergence puzzle — began in response to the rising need for a common automation platform for different network lifecycle processes. The goal for the project is to leverage and integrate existing next-generation network technologies to highlight the importance and novelty of converged service operations. We envision CSML acting like a master element management system (EMS) which communicates southbound to various domain-specific infrastructure layers, EMS, network functions (physical and virtual) in order to develop converged services. Typically, domain-specific management systems, such as for hybrid fiber-coaxial (HFC) or mobile networks, do not have the visibility or control over other domains for FCAPS tasks, and a central management entity such as CSML can enable that inter-domain communication to achieve multi-access convergence. By supporting management mechanisms of various domains, CSML can act like a single point of control that allows operators to employ virtual functions alongside physical functions and utilize their different network domains more coherently.

The term service orchestration is oftentimes used to describe distinct concepts in the industry. In addition, terms like workload orchestration or network orchestration are also employed in similar but different contexts. Workload orchestration typically refers to installing an application and its associated components over a single target virtual infrastructure, such as using OpenStack for virtual network functions (VNFs) or Kubernetes for cloud-native network functions (CNFs). It is predominantly limited to the management of a single cluster running its own control and user planes. Network orchestration refers to managing network configurations and policies for physical or virtual functions through a single pane of glass, typically employing a SDN controller for centralizing the control planes to manage multiple user planes. Service orchestration operates at a higher layer than workload and network orchestration and leverages them to deploy and manage services comprising of multiple applications to be installed on different target infrastructures, while integrating all control functionalities into a single framework. It uses workload orchestration to instantiate network functions as VNFs or CNFs over various target infrastructures and uses network orchestration to configure physical and virtual networks spanning multiple domains. Figure 1 highlights the difference between different layers of orchestration. Each domain, such as cable access and core, or mobile RAN and core, generally employ domain-specific orchestrators and controllers. For a converged service encompassing multiple domains and utilizing separate workload and network orchestrators, a service orchestrator such as CSML can help integrate all the different components to provide true operations convergence.

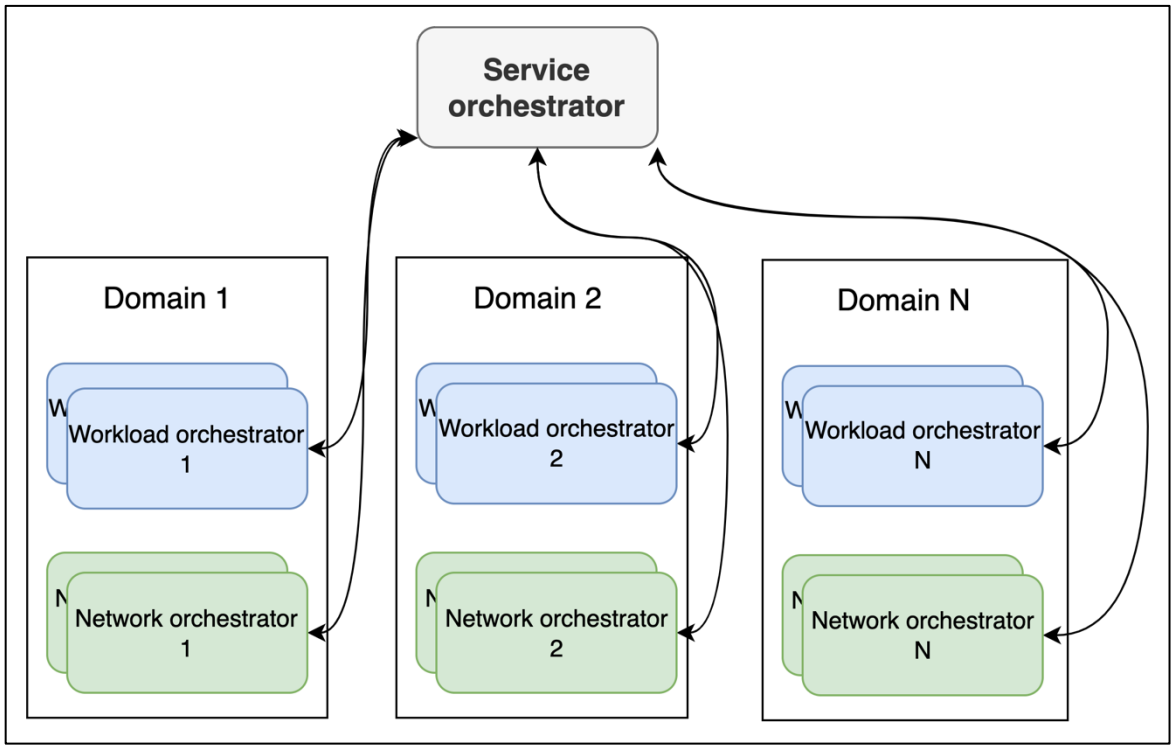

Figure 1 - Service orchestration as compared to workload or network orchestration

During the runtime of the project, several different use cases were developed and implemented targeting different network domains and lifecycles. The first use case targeted virtualized functions in the mobile and Wi-Fi domains, while the second use case focused on incorporating HFC-domain physical functions into the framework to demonstrate hybrid orchestration. This paper presents details on the second use case developed on HFC networks – dynamic speed boost - to demonstrate the value of operations convergence.

Typically, in cable networks, the process of requesting for and implementing a change in subscriber bandwidth or QoE is manual and requires a reboot of the cable modem (CM) for the change to take effect. The PacketCable Multimedia (PCMM) technology provides a mechanism to create dynamic quality of service QoS policies by leveraging functionalities defined in the DOCSIS and PacketCable DQOS specifications [1]. We onboard PCMM into CSML to enable dynamic, closed-loop, application-specific QoS control over the legacy CMTS to support existing and future QoS-enhanced services.

This paper presents details on the overall PoC architecture involving CSML, physical network functions (CMTS and CM) and a virtual PCMM domain controller, and discusses the end-to-end technicalities of the PoC. The rest of the paper is organized as: section II presents details on CSML, its different building blocks, and describes the first use case briefly, section III provides details of the dynamic cable speed boost PoC developed, and section IV presents the conclusion and directions for future work.

2. Converged Service Management Layer

The European Telecommunications Standards Institute (ETSI) Industry Specification Group for NFV (ETSI ISG NFV) developed a framework for NFV management and orchestration (MANO) of all resources in a virtualized data center [2]. Figure 2 shows the NFV MANO framework and the different components managing different network layers. The virtualized infrastructure manager (VIM) communicates with the NFV infrastructure (NFVI) layer to manage the underlying virtualized resources – virtual compute, virtual networking, and virtual storage (such as OpenStack or Kubernetes). The VNFM (VNF manager) interacts with both element managers (EM) and VNFs directly to manage the fault, configuration, accounting, performance, security (FCAPS) of individual VNFs. The NFV orchestrator (NFVO) interacts with OSS/BSS and the underlying management entities (VNFM and VIM) to manage the lifecycle of complex services comprising of multiple VNFs.

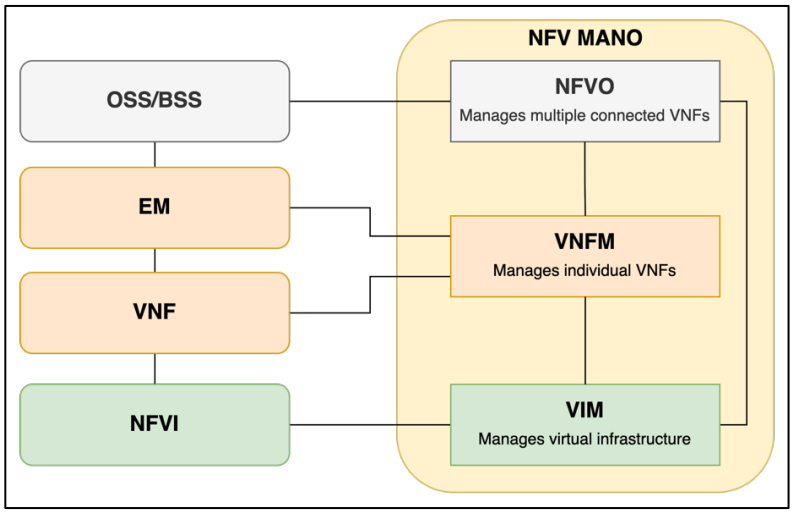

Figure 2 - NFV MANO framework [2]

In the context of different orchestration layers, the NFVO is a subset of the service orchestrator for NFV infrastructure, the VNFM maps to the workload orchestrator managing the lifecycle of individual functions, while the VIM compares with the network orchestrator providing underlying network connectivity. Since DOCSIS, optical, and mobile networks all include some functions implemented in specialized hardware, CSML extends the NFV MANO framework to include the management of physical network functions (PNFs) as well. Figure 3 illustrates the enhanced MANO framework employing CSML to manage PNFs as well. CSML acts as the NFVO and VNFM components from the NFV MANO framework. It includes the

capability to manage VNFs using either its in-built generic-VNFM (gVNFM) or by interacting with specific-VNFMs (sVNFM), including an interface to the VIM layer for virtual resource management over the NFVI. Similarly, CSML can manage PNFs either through specific EMs or by directly interacting with PNFs using its PNF manager (PNFM) over well-known management protocols.

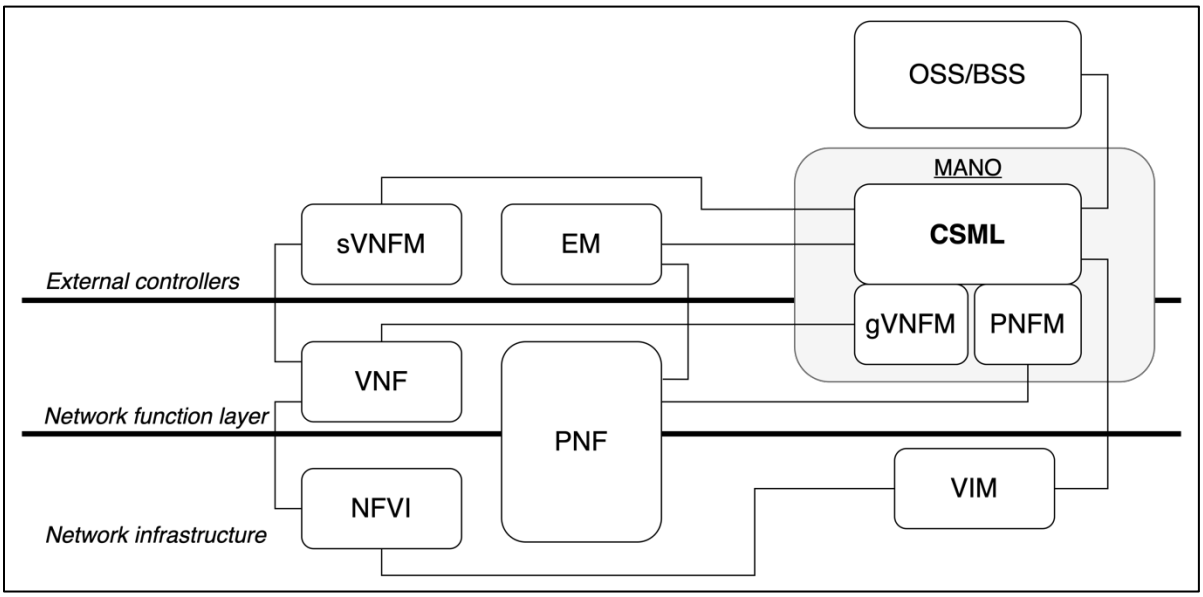

Figure 3 - Enhanced MANO framework using CSML for VNF and PNF management

On a high-level, as illustrated in Figure 4, CSML enables three types of lifecycle management (LCM) activities as a part of its two primary frameworks, namely the design-time framework and the run-time framework:

1. Service design: Involves composing services comprising of multiple xNFs, along with their Day-0 and Day-N configuration parameters and policy rules to enable elastic management of the service.
2. Service deployment: Involves instantiating the modeled services on to the target infrastructures (both physical and virtual) and supervising scale-in/out as needed.
3. Service assurance: Involves monitoring the deployed services and taking closed-loop actions using analytic tools to make the framework self-healing and self-optimizing.

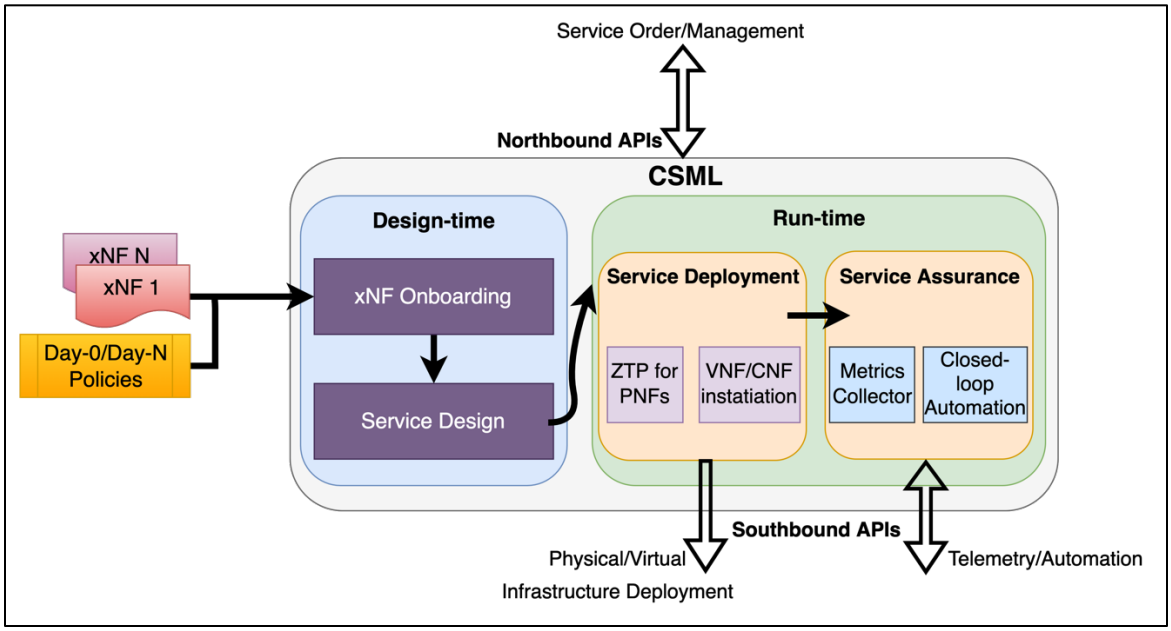

Figure 4 - High-level CSML architecture

The first use-case developed as part of the project activity was to orchestrate and assure a wireless steering application [3]. This application provides an over-the-top solution to steer user traffic over Wi-Fi or mobile network based on current network performance or operator requirements. It is a client-server-based application with the client running on user's mobile device and the server running in the cloud. The server service comprised of eight containerized network functions which needed to be running on different cloud locations simulating an operator's production network setup. This use case highlighted the value of using a service orchestrator such as CSML that can interact with multiple underlying workload orchestrators to deploy a distributed service over different cloud locations comprising of both public and private clouds.

Figure 5 illustrates the network topology developed for this use case. The different server components of the application are deployed on different cloud regions to showcase the capability of CSML to deploy network functions based on their network and compute requirements. For example, the control gateway is an edge component and needed to be geographically closer to the user, therefore it was deployed on Azure Central region, while the data gateway is the data path component through which all user traffic traverses, and therefore it is deployed on a private cloud. The VIM layer consisted of Kubernetes-based providers – Azure Kubernetes Service (AKS) for the public cloud locations and RedHat's OpenShift for the private cloud location. CSML itself was implemented to run in AKS, acting as the service orchestrator and was able to demonstrate multi-cloud service orchestration involving instantiating components over different locations, interconnecting them over virtual networks, and ensuring the end-to-end service operated as expected.

CSML was also responsible for assuring and healing the service by continuous monitoring and taking a specific closed-loop action when an issue was detected. The scenarios tested included - *(i)* Application-level healing – The control gateway component was responsible for leasing out IP addresses from a pre-defined IP pool to UEs connecting to the steering application, and as the IP pool would start to get exhausted, no new UEs would be able to connect to the application. CSML was able to identify this issue by continuously monitoring the control gateway's runtime via one of its REST endpoints and as 90% of the pool became exhausted, a new IP pool was allocated to by invoking another REST endpoint and modifying its Day-N configuration to ensure service continuity; *(ii)* Infrastructure-level healing – Since the control

gateway component is running on a shared cluster along with other services, there could be instances when the underlying compute resources become deficient. CSML was able to identify this issue by monitoring cluster resources via Kubernetes APIs, and resolving it by migrating the control gateway to another cluster and interconnecting all other components to the new cluster to ensure no service disruption.

Furthermore, additional utilities were developed in order to enable federated collection of telemetry data of both the infrastructure-level metrics as well as application-level metrics in order to enable centralized decision-making. This use case proved the value of service orchestration for virtualized environments spanning multiple locations and provided a common, abstracted framework for deploying and managing novel services.

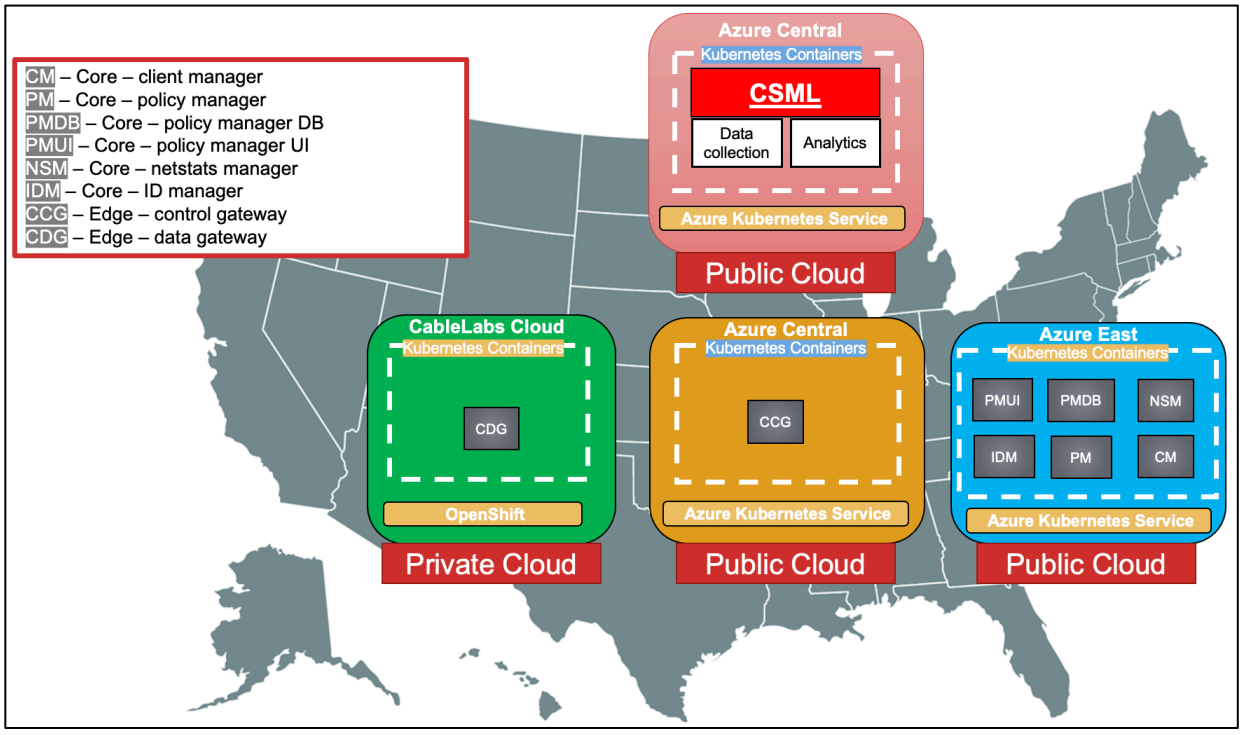

Figure 5 - Multi-cloud service orchestration using CSML

3. Dynamic Cable Speed Boost

The second use-case developed, to demonstrate hybrid physical-virtual orchestration, was to use CSML to enable dynamic speed boost in the cable network. DOCSIS provides two mechanisms to configure service flows (QoS policies) – *(i)* Static, which are established during the CM registration process, and *(ii)* Dynamic, which can be configured on an as-needed basis. Typically, operators employ the first mechanism which entails pushing a new configuration file to the CM manually and requires a reboot of the CM for it to take effect. This process is relatively lengthy and leads to disruption in traffic for the time it takes the CM to boot back up. The dynamic service flow mechanism requires an entity to interact with multiple controllers and network devices in order to translate the required QoS request into configurable DOCSIS service flows. CSML can act as the central orchestrator in this scenario to realize this use case by leveraging its capability to control both virtual and physical functions.

Figure 6 illustrates the high-level network topology of the use case implemented. The dynamic speed boost app is a web-based GUI developed using the Python-Flask framework, running in a VM in public cloud, which provides the user with the choice to boost or un-boost a specific application for a definite or indefinite time [4]. The application options included as part of this use case include YouTube (classified based on TCP port 443), Zoom audio (classified based on DSCP value 56), and Zoom video (classified based on DSCP value 40). Additional application options could be included based on requirements and differentiating parameters. CSML exposes northbound REST APIs which the dynamic speed boost app invokes to pass the user requested QoS parameters. In the southbound direction, interfaces were developed for CSML to connect to a PCMM controller and a CMTS in order to create, modify or delete service flows and verify if they have been pushed to the CM, respectively. OpenDaylight (ODL) was used as the PCMM controller in this use case as it is open-source and includes an implementation of the PCMM service as an additional module [5]. ODL was also running as a VM in public cloud. ODL exposes northbound REST APIs to provision a CMTS and service flows, and uses the Common Open Policy Service (COPS) protocol as transport to communicate with the CMTS. CMTS uses MAC-layer DOCSIS Dynamic Service Add/Change/Delete (DSA/DSC/DSD) messaging for service flow management for a specific CM. A Cisco cBR-8 was used as the CMTS and an Arris SURFboard model was used as the CM. A laptop was connected behind the CM for speed-testing purposes.

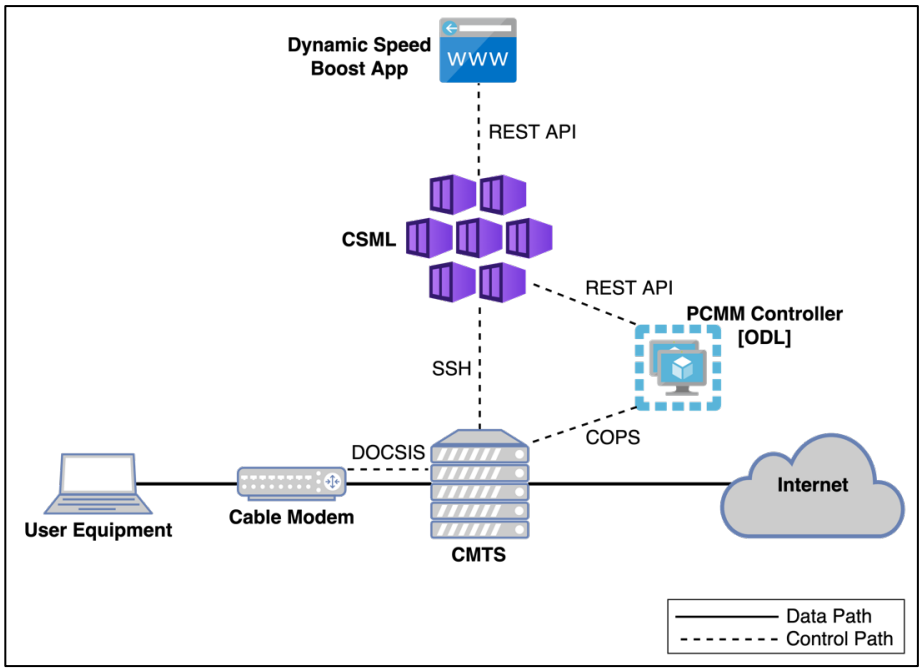

Figure 6 - Network topology of dynamic cable speed boost

The design-time work within CSML involved creating blueprint models for ODL and CMTS in order to enable their Day-N configuration. The network function (NF) model is essentially an archive consisting of TOSCA-based definitions represented in JSON, and specifications for run-time actions. The blueprint definitions include reusable data dictionaries, and component and node workflows to allow for intent-based configuration of the NFs. The run-time model defines the southbound interface and the associated actions that can be invoked to automate NF configuration. The southbound interface used to configure ODL was a REST-based environment that enables pipelining multiple API request-responses defined in a workflow, while the southbound interface used for the CMTS was a Python-based environment that allowed reusing existing scripts to SSH into the CMTS and change configuration. The Python-SSH script employed the

Netmiko library to set up connections with the CMTS, and send and receive commands related to service-flow configurations [6].

The run-time work within CSML involved developing northbound APIs and defining their associated payloads that could be used by an external application, the dynamic speed boost app in this scenario, in order to request for a boost for a particular application. CSML translated this request and invokes a defined workflow to transfer the requests to the correct run-time environments and take corresponding actions based on responses received from the NFs. This process enables an abstracted service framework which can be used by external northbound systems to interact with CSML, and in turn the underlying NFs, without knowing specific details about individual NFs.

Figure 7 depicts the multiple interactions of CSML with different NFs. The end-to-end flow works as:

1. User requests for a speed boost by logging into the dynamic speed boost app and providing details of the application requiring boost and an optional end time.
2. The request gets communicated to CSML over its northbound REST API including the CM IP of the requesting user.
3. CSML invokes a workflow defined for this purpose that includes the steps to fulfil this request.
4. CSML first checks if the COPS connection between the CMTS associated with the requesting CM and ODL is up. If it is not, CSML initiates the connection by sending an API request to ODL.
5. Next, CSML communicates the service flow request to ODL consisting of the CM IP, classifier to match specific application traffic, and either a pre-configured service class name on the CMTS or DOCSIS QoS parameters to define the speed boost requested.
6. ODL uses COPS protocol to communicate with CMTS the new service flow required.
7. CMTS uses DOCSIS to push the service flow to the specific CM.
8. Once ODL returns the confirmation to CSML that the flow has been pushed, CSML initiates a SSH connection directly to the CMTS and requests all service flows active for that CM IP.
9. CMTS returns the list of active service flows to CSML.
10. CSML verifies that the new service flow pushed matches the original request from the user.
11. After verification, CSML returns the boost confirmation message to the dynamic speed boost app over its northbound REST API.

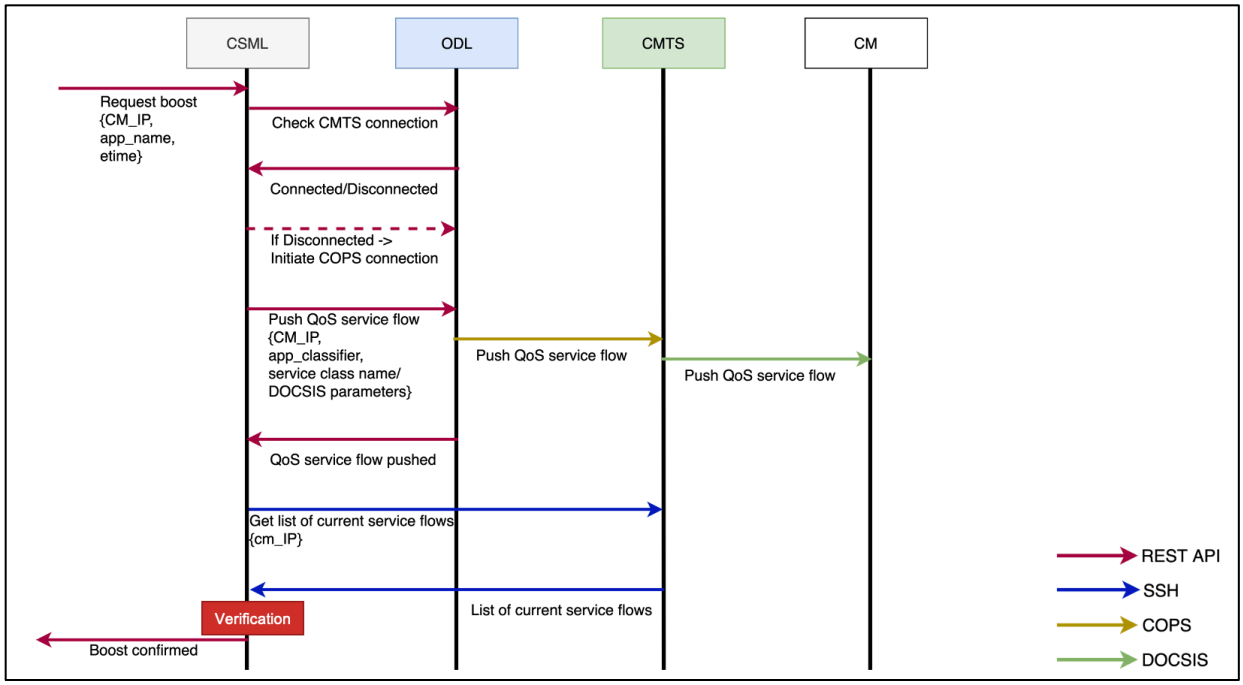

Figure 7 - End-to-end interactions of CSML with different NFs

The end-to-end flow described above illustrates the level of abstraction offered by using a converged service orchestrator. The OSS/BSS or external applications are only required to send a single request to CSML, and all the non-trivial interactions with different NFs, both physical and virtual, are implemented on the backend within CSML to fulfil the request. This enables integrations with multiple different NFs and controllers, and allows for the development of novel use cases with the orchestrator handling much of the operational complexities associated with them.

To test the overall functioning of this use case, the user laptop was used to test different application speeds before and after requesting for the boost. The user laptop was running a speed test application, YouTube application, and an iPerf client which was connected a public iPerf server on TCP port 5002. The default subscriber bandwidth was set to 10 Mbps for both upstream and downstream in the CM configuration file. Next, a 4K video was opened on the laptop and real-time video statistics were enabled to check connection speeds for the running stream. Initial speeds were capped at 10 Mbps, which resulted in excessive video buffering, and the iPerf speeds were also capped at 10 Mbps for both upload and download. To improve video performance, the user then requested for a speed boost of the YouTube application by logging into the dynamic speed boost app. Almost in real-time, CSML was able to fulfil this request and pushed a new downstream service flow (since YouTube video streaming is primarily downstream data-driven), and the video statistics indicated connection speeds going up to ~85-90 Mbps now, with the video no longer buffering anymore, as shown in Figure 8. At the same time, iPerf download speeds were still capped at 10 Mbps since it was running on a different port as that of the requesting application. This demonstrated the capability of CSML to achieve a dynamic cable speed boost for a specific application based on user requirements.

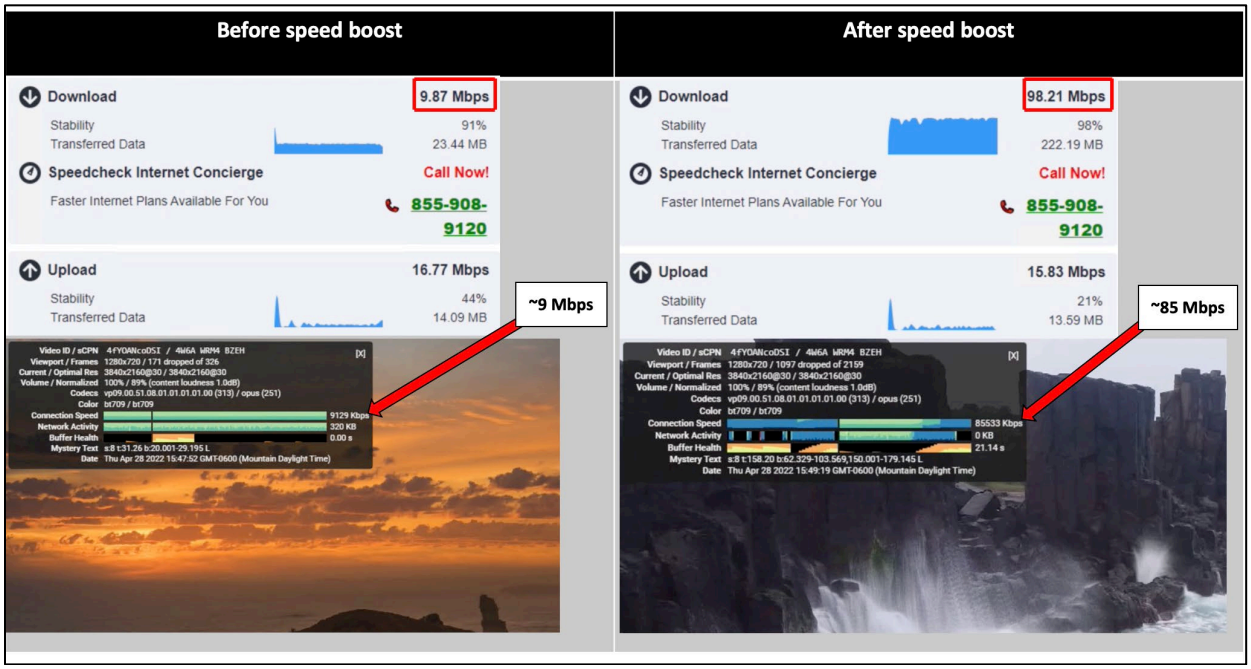

Figure 8 - Speed test results and 4K video connection speeds before and after speed boost

Although, individual scripts could also be used to achieve the same use case, however, orchestration provides additional value to the process. Firstly, developing scripts to communicate with different NFs is not trivial, CSML includes a run-time framework that removes much of the complexity to communicate with NFs and controllers, and simplifies creating complex workflows with multiple interactions. Secondly, once a NF or controller has been modeled within CSML, the same run-time environment can be extended to similar devices running the same management protocols. For example, while in this scenario a Python-SSH and a REST-API environments were developed to communicate with the CMTS and ODL controller respectively, the same could be leveraged to manage switches/routers over SSH and other controllers exposing REST APIs. Thirdly, once a NF or controller has been onboarded within the orchestrator, it can be reused for other use cases to create complex service chains employing different xNFs.

4. Conclusion

With the proliferation of virtualized network functions and the need to converge the management and operations of different network domains, a centralized service orchestrator is required for integrating different systems, thereby enabling novel use cases while reducing operational complexities. This paper introduced the CSML framework along with specific use cases developed to demonstrate the value of service orchestration and automation. The dynamic cable speed boost use case is described in detail which allows for changing the cable subscriber bandwidth on the fly. Both physical and virtual NFs and controllers were onboarded within CSML, and specific southbound run-time environments were leveraged in order to create a complex service chain including a PCMM controller and CMTS. Speed testing was performed to compare different application speeds before and after the user requests for a speed boost, and the results indicated that granular, real-time changes to subscribed bandwidth are achievable in DOCSIS networks using an intelligent, converged orchestrator framework such as CSML. The broader goals of the CSML project are to drive the adoption of network automation, virtualization, and operations convergence at scale. Also, as the transition to NFV is progressing, the project aims to demonstrate how physical network elements can be harmonized with virtual elements to preserve exiting network investments.

While the use cases described in this paper demonstrated both physical and virtual orchestration, they were limited to domain-specific architectures. The next phase of the project is to develop a true converged use case that involves incorporating 5G core into the framework in order to provide additional value to operators that provide both cable and mobile services. For example, the cable speed boost use case could be extended by – *(i)* Identifying customer devices that are subscribed to both cable and mobile services from the same operator and, *(ii)* Boosting their Wi-Fi bandwidth to offer more value and help differentiating from other operators.

Abbreviations

AKS	Azure Kubernetes Service
CM	cable modem
CMTS	Cable Modem Termination System
CNF	cloud-native network function
COPS	Common Open Policy Service
COTS	commercial off-the-shelf
CSML	Converged Service Management Layer
DOCSIS	Data-Over-Cable Service Interface Specification
EMS	element management system
ETSI	European Telecommunications Standards Institute
FCAPS	fault, configuration, accounting, performance, security
HFC	hybrid fiber-coaxial
MANO	management and orchestration
NF	network function
NFV	network functions virtualization
NFVI	network functions virtualization infrastructure
NFVO	network functions virtualization orchestrator
ODL	OpenDaylight
PCMM	PacketCable Multimedia
PNF	physical network function
PNFM	physical network function manager
QoS	quality of service
SDN	software-defined networking
VIM	virtualized infrastructure manager
VM	virtual machine
VNF	virtual network function
VNFM	virtual network function manager

Bibliography & References

[1] Cable Television Laboratories, Inc., “PacketCable™ Multimedia Specification”, PKT-SP-MM, November 2019. [link]

[2] European Telecommunications Standards Institute, "Network Functions Virtualization (NFV); Management and Orchestration; Architectural Framework Specification", ETSI GS NFV, June 2022. [link]

[3] Cable Television Laboratories, Inc., "IWiNS—An Informed Approach to Mobile Traffic Steering", January 2021. [link]

[4] The Pallets Projects, "Flask - The Python micro framework for building web applications". [link]

[5] The Linux Foundation Projects, "OpenDaylight". [link]

[6] Kirk Byers, "Netmiko - Multi-vendor library to simplify CLI connections to network devices". [link]

