Unleash the Power of Cloud Computing for CMTS

A Technical Paper prepared for SCTE by

John Chapman
CTO Broadband Technologies, Cisco Fellow
Cisco Systems Inc.
170 W Tasman Dr, San Joe, CA 92677
408-526-7651
jchapman@cisco.com

Tong Liu, Ph.D
Principal Engineer
Cisco Systems Inc.
300 Beaver Brook Road, BOXBOROUGH, MA 01719
978-936-1217
tonliu@cisco.com

Table of Contents

Title **Page Number**

List of Figures

Title **Page Number**

1. Introduction

Not so long ago, the CMTS was all about the hardware, the chassis, the cable line cards with embedded CPU, memory and storage. It had long development cycles and expensive/lengthy processes for deployment and upgrade. The virtualization of the CMTS removed the need of the specialized hardware by providing the CMTS services in software running on generic servers. By doing so, it has greatly improved the efficiency of the CMTS hardware infrastructure and started the cloud native transformation of the CMTS software.

With the virtualization laying out the foundation, moving the CMTS to the cloud is a natural step forward to scale the CMTS beyond the on-premises physical servers. Cloud provides on-demand compute, memory and storage resources with high scalability, availability and security. It offers users the economies of scale and better cost efficiency with a pay-as-you-go cost model. It reduces the application development time and simplifies the deployment/upgrade processes by managing the underneath infrastructure and platform on behalf of the users.

CMTS cloudification is not just a rehosting effort, it is an optimization process that requires proper service decoupling and workload placement. The cloud computing services can be “resource-aware” or “serverless”. The resource-aware service, such as Amazon Elastic Compute Cloud (EC2), allows users to control the resource allocations and autoscaling at a relatively coarse granularity.

The serverless service, such as AWS Lambda or Amazon DynamoDB, fully manages the resources on behalf of the users with built-in fine-grained auto scaling capabilities. The geographical location of the cloud platform also matters as it impacts the network latency and the data transport cost. The distributed and inhomogeneous cloud environment presents an interesting and challenging problem for the cloudification of the CMTS.

In this paper, we explore the cloudification solutions for the CMTS. Following the introduction, we first overview the vCMTS structure in Section 2 and the public cloud infrastructure in Section 3. We then explain the cloudification motivations in Section 4 and describe the different architecture choices in Section 5. We examine the different cloud hosting options in Section 6 and explore the CMTS service placement strategies in Section 7. In section 8, we show an example for deploying the CMTS services on AWS. In Section 9, we explore how to take advantage of the serverless platform by converting the stateful DOCSIS processes into stateless functions. In Section 10, we discuss how to start the cloudification with a proof-of-concept (PoC) effort. In Section 11, we study how to deliver CMTS as a service by using CI/CD and the cloud powered multi-tenant architecture. Finally in Section 12, we conclude the paper.

To simplify the text and precisely describe the intended cloud entities, we use the Amazon Web Services (AWS) terminologies in this paper and provide links to AWS original definitions in the Reference Section. Similar terminologies and definitions can be easily found on other public cloud providers’ platforms such as Microsoft Azure, and Google Cloud platform,

2. vCMTS Architecture Overview

The vCMTS virtualizes the physical CCAP core in the DOCSIS distributed access architecture (DAA) [1] as shown in Figure 1. It is typically installed in a headend or hub that is connected to both the converged interconnect network (CIN) and the service provider (SP)’s core network. Function wise, the vCMTS contains the following service domains with distinctive timing requirements:

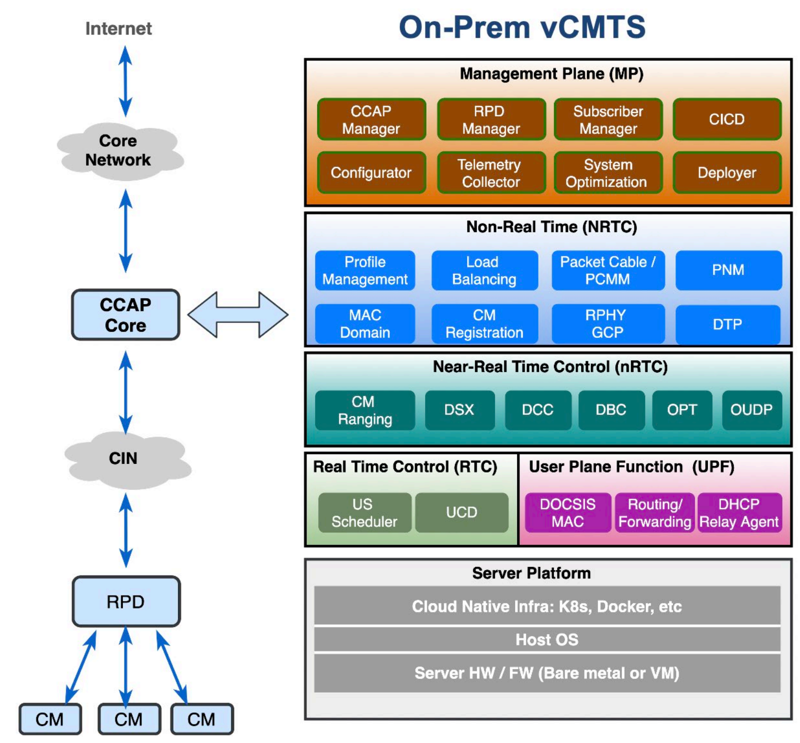

Figure 1 – vCMTS in Cable Distributed Access Architecture

Management Plane (MP)

MP contains the non-real-time (latency > 1sec) management functions to control/optimize the CMTS network elements, for deployment, configuration, health monitoring and carrying out any corrective actions if needed.

Non-Real Time Control (NRTC)

Non-real time (latency >1sec) control, NRTC, refers to the CMTS control functions that have a timing budget of 1second or above, such as CM registration, load balancing, profile management, and proactive network management etc.

Near-Real Time Control (nRTC)

Near-real time (10ms – 1sec) control, nRTC, refers to the CMTS control functions that have a timing budget of a few hundreds of milliseconds, such as DOCSIS ranging, DSX and DBC etc.

Real Time Control (RTC)

Real-time (<10ms) control, RTC, refers to the CMTS functions that have a timing budget of single-digit milliseconds, such as DOCSIS upstream scheduling and functions, UCD update transaction and MAP replications.

User Plane Functions (UPF)

The term of UPF is borrowed from mobile for describing the data plane functions that require high-throughput and low latency, such as DOCSIS MAC frame processing, packet routing/forwarding, and DOCSIS downstream scheduling.

The latency values for RTC, nRTC, and NRTC are adapted from the Open Radio Access Network (ORAN) Alliance [2].

The vCMTS today is hosted by a dedicated server or a server cluster on premises that contains the compute, memory, storage and networking resources, and certain server/platform applications, such as Docker, Kubernetes, Radis, Kafka etc., to run the containerized CMTS microservices.

3. Public Cloud Infrastructure

Public cloud delivers virtualized computing services on-demand globally through internet and/or direct connections. Today the major public cloud platforms have practically covered all locations that have a critical mass of Internet users.

Depending on the distance relative to the end users, the public cloud infrastructure can be either in the Region or at the Edge using the AWS terminologies, defined as below.

Region

A Region is a physical location around the world that contains multiple data centers within a geographic area, such as US East (Northern Virginia), US West (Northern California) etc. Each Region consists of multiple, isolated, and physically separate Availability Zones (AZs).

Each AZ has one or more discrete data centers with redundant power, networking, and connectivity in a Region. Within the Region, an AZ is physically separated by a meaningful distance, many kilometers, from any other AZ, although all are within 100 km (60 miles) of each other.

Edge

The Edge infrastructure provides the computing services as close to the endpoints as necessary, deployed as the cloud-managed hardware and software in locations outside the Region and even onto the customer owned devices themselves. Edge is typically used by latency sensitive or data intensive applications which would benefit from the local processing for avoiding the latency or network cost for shipping the workload to the Region.

AWS edge offers multiple edge computing solutions including AWS Outposts for on-premises, AWS Local Zones for metro areas, and AWS Wavelength for the 5G Edge.

Figure 2 shows the AWS Region map and edge networks in North America [3]. Similar coverage can also be found on other public cloud platforms. For a vCMTS, both the Region and the Edge may be used for cloudification.

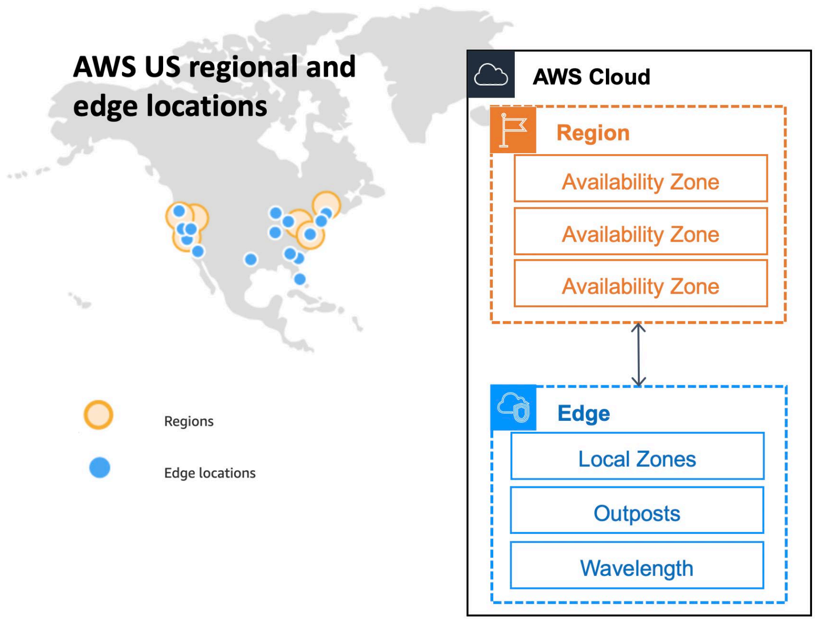

Figure 2 – AWS Infrastructure and Region Map in North America (2021)

4. Moving the vCMTS to Cloud

Just like virtualization removes the need for the specialized hardware, cloudification aims to remove the need for owning the physical servers, instead use the cloud compute services with a consumption-based cost model.

Moving to the cloud benefits both the CMTS vendors and the operators. For the CMTS vendors, cloudification simplifies the infrastructure/platform development, so they could have more time/resource innovating on their unique market differentiators. For the CMTS operators, cloudification promises global coverage, built-in high availability and rapid scaling capabilities. With on-demand cloud computing, operators have less risk for over-provisioning or under-provisioning, no need to worry about upfront costs and on-going server maintenance (provided by the cloud provider), or patching and software upgrades (provided by vendor).

The chosen public infrastructure / platforms should best suit the CMTS workloads in terms of proximity, performance and cost efficiency. Hybrid and multi-cloud strategy may be used to adapt to specific CMTS deployment needs.

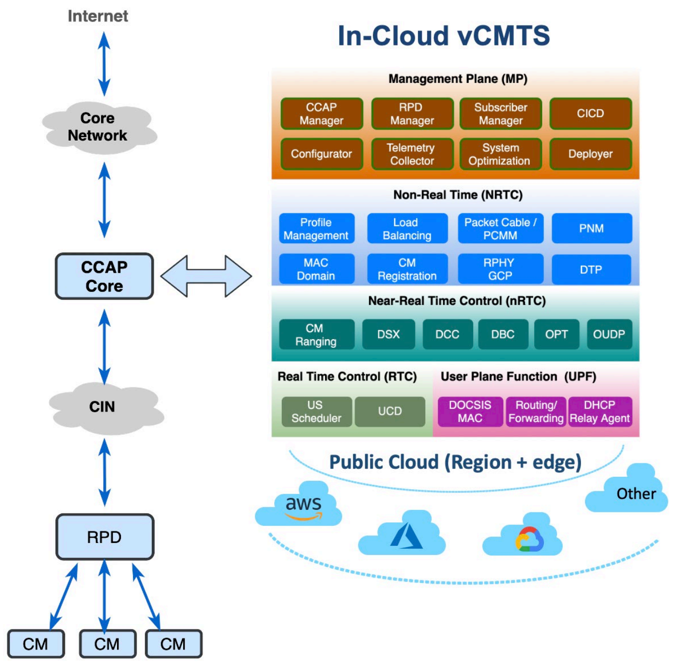

Figure 3 – vCMTS Hosted in Public Cloud

5. Cloudification Archtecture Choices

Migration to the cloud is hardly a one-step move. It involves multiple iterations to achieves the optimum efficiency. Figure 4 shows the typical architectural choices that may be used along this journey.

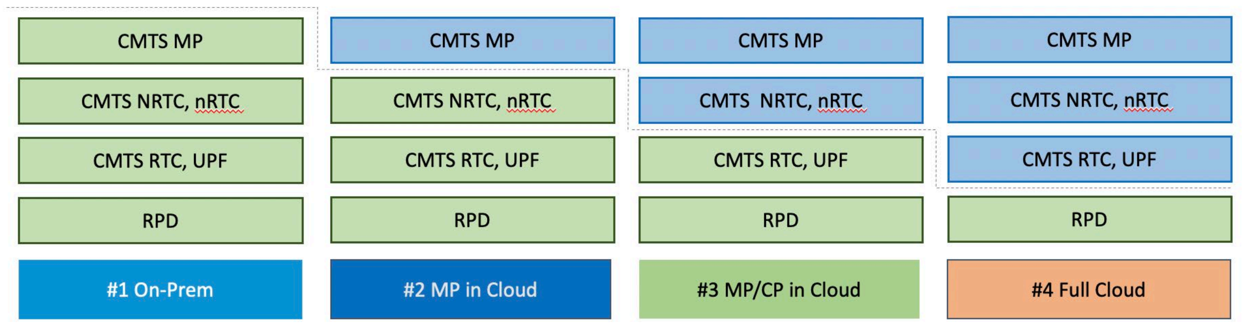

Figure 4 – CMTS Cloudification Choices

#1 On-Premises

This is the vCMTS with all the CMTS service domains hosted on premises and no cloud involved. This represents the majority of vCMTS deployments today.

#2 MP in Cloud

By simply moving the MP to the cloud, this could be a phase-one approach for cloudification. It also accomadates the FMA DAA where DOCSIS is built in the Nodes attached to the plant.

#3 MP and CP in Cloud

This is a hybrid development/deployment environment involving both cloud computing and the private on-premise compute platform. Specifically, the MP and the DOCSIS control plane (CP), including the NRTC and nRTC functions, are in the cloud, while the UPF and the RTC functions remain on-premise. This option requires that the DOCSIS control and user planes are separated with proper interface definitions.

#4 Full Cloud

The CCAP core is fully moved to the cloud. More details can be found in Section 8 where a full cloud deployment example is presented using both the Region and the Edge platforms. A Region-only full cloud setup can be found in Section 10 to test drive the cloud computing platform. This scenario applies if the cloud provider is providing Internet transit services or if the vCMTS is paired with a test harness that includes virtual DAA nodes and virtual CMs.

6. vCMTS Hosting Options

The cloud environment offers a variety of hosting options for the vCMTS services, as shown Figure 5. This includes the options of physical servers on premises, a “resource-aware” virtual server platform, or a “serverless” platform.

The vCMTS today typically uses the physical servers on premises, owned and managed by the service provider. It scales statically by the number of physical servers, with each server as a pooling unit for all resource types including compute, memory, storage and network interfaces. Once installed, the vCMTS capacity is fixed regardless the actual workloads. Provision is typically needed based on the workload in the worst case scenario, resulting in wasted resources during idle time.

The on-premises hosting option does have its advantages for handling the latency sensitive and data intensive workloads, such as UPF and RTC. Because of this, all major cloud platforms have the on-premises extensions, either as physical servers, such as AWS Outposts, or software agents/packages running on the user-owned servers.

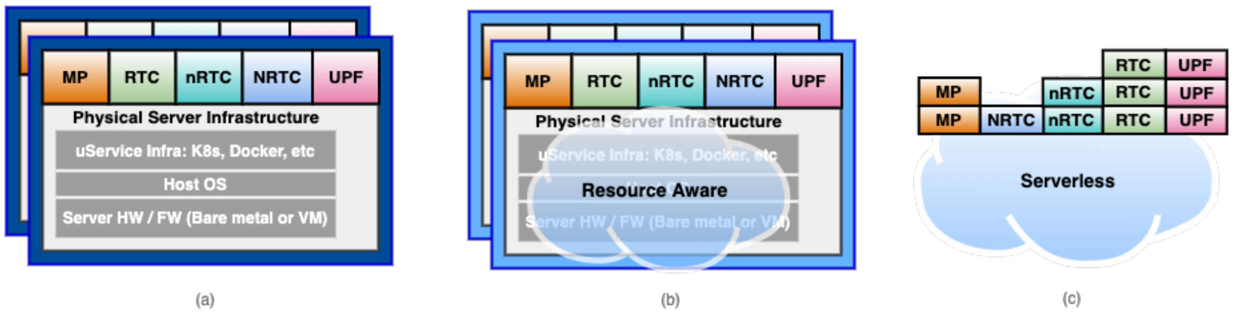
MP
RTC
nRTC
NRTC
UPF
Physical Server Infrastructure
uService Infra: K8s, Docker, etc
Host OS
Server HW / FW (Bare metal or VM)
(a)
MP
RTC
nRTC
NRTC
UPF
Physical Server Infrastructure
uService Infra: K8s, Docker, etc
Resource Aware
Server HW / FW (Bare metal or VM)
(b)
RTC
UPF
MP
nRTC
RTC
UPF
MP
NRTC
nRTC
RTC
UPF
Serverless
(c)

Figure 5 – vCMTS Server Platform Options

The “resource-aware” platform is the traditional cloud computing platform provided by virtual machines (VMs) overlaid on physical hardware, such as Amazon EC2 [4]. On this platform, users can configure the VM instances and organize clusters without the need to own the underneath physical server. In fact, they only rent a fraction of the cloud infrastructure to run the workload and achieve the economies of scale of sharing the infrastructure with others. Since users can control the resource type and running time, this platform is suitable for long-lasting or stateful workloads. However, there is the overhead to configure the VMs and the autoscaling is in the number of VMs, a coarse granularity comparing to the next serverless platform.

“Serverless” is a new cloud computing technology that keeps the infrastructure/platform completely hidden from the user [5]. It aims to simplify the application development/deployment processes, and enables a fine-grained, transaction-based consumption model. The serverless resources are automatically scaled on the user’s behalf as the workload changes. The serverless platform started with AWS Lambda, a function-as-a-service platform, and later expanded to include any resource-unaware backend services, such as Amazon DynamoDB for database, Amazon Simple Storage Service (Amazon S3) for binary storage, and AWS Fargate for running the containers.

The serverless platform is rapidly evolving but is comparatively less mature. For example, it is limited today in hosting stateful or latency sensitive applications, as the underneath compute resource is ephemeral and invisible to user applications. To take advantages of serverless platform, applications need to be stateless, event driven and able to tolerate the invocation delays.

7. CMTS Workload Placement Strategies

CMTS is a complex system, producing a wide variety of workloads that have diverse requirements on the infrastructure/platform, such as the distance/latency to the cable modems (CMs), processing speed, memory size, I/O performance, and CPU types etc. On the other hand, the cloud environment is distributed and inhomogeneous in nature. Having an optimum workload placement is critical to minimize cost, ensure the CMTS performance.

Figure 6 shows the mapping of the key workload requirements (y-axis) to the available cloud platform types (x-axis). It reveals the suitability for placing a CMTS workload onto a specific platform, as described below:

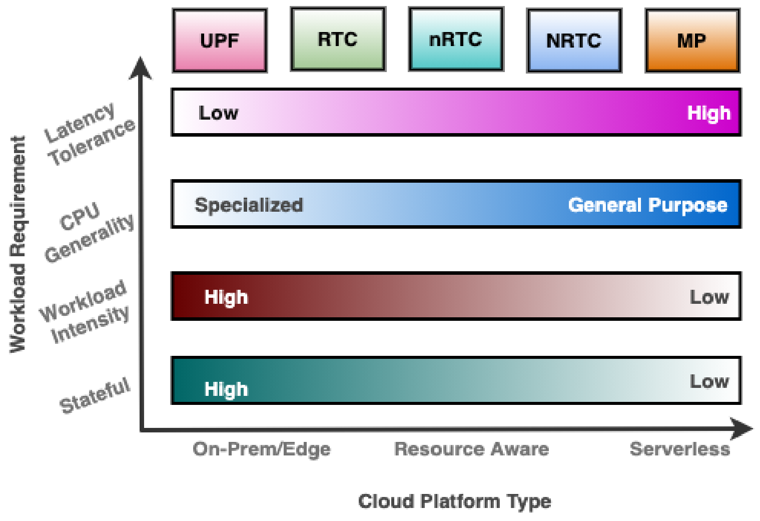

Figure 6 – CMTS Workload Placement Strategies

UPF and RTC

The UPF and RTC workloads are suitable for the on-premises/edge platform, as both types are latency sensitive and/or data intensive.

The UPF workloads are time sensitive and data intensive, up to millions packets per second per Service Group (SG). The RTC workloads are timing sensitive and compute intensive. The DOCSIS US scheduling decisions need to be made every millisecond with a processing time limited to the low hundreds of microseconds. Accurate DOCSIS timing is also required to align MAPs with the US physical layer. The RTC relies on the UPF for forwarding the real-time signaling traffic. The combined platform service access delay must not exceed the overall latency budget of low single digit of milliseconds.

nRTC

The nRTC workloads are the DOCSIS control protocols with a time budget of a few hundreds of milliseconds, including the signaling propagation delay and the processing delay. Examples of the nRTC workloads include ranging request handling, dynamic service flow addition/change/deletion (DSX), and dynamic bonding group change (DCC). The nRTC timing requirement is more relaxed than the UPF and RTC requirements, but tighter than the NRTC and MP workloads.

As an optimization strategy, the nRTC workloads can be placed in the Region to free up the Edge resources for UPF and RTC. Since the in-region serverless platform today have non-negligible invocation delay that may exceed the nRTC timing budget, the in-Region resource aware platform, such as Amazon EC2, may suitable for the nRTC workloads.

NRTC and MP

The NRTC workloads, such as CM registration, MP workloads, and as SG configuration, are typically infrequent and latency tolerant (minimum one second). Occasional usage spikes may occur, for example, during a planned upgrade/maintenance window or upon an unexpected power outage. In which case, the NRTC may experience a CM registration storm with potentially thousands of CMs trying to register at the same time.

These characteristics match well with the serverless platform, where the NRTC and MP would benefit from the built-in elasticity and the economics of the fine-grained cost model. During the normal operation time, the serverless platform may not incur any cost as the NRTC or MP workloads are very low and may well be within the free tier of the serverless services. During the rare event like the CM registration storm, the serverless platform can rapidly scale up to meet the demand and scale down once as storm fades away.

In summary, the CMTS workload placement is an optimization process to minimize cost within the boundary conditions for the CMTS to perform at a given scale.

8. Full Cloud CMTS Deployment Model

A full cloud CMTS may be deployed on both the edge and the Region for an optimum workload placement. Figure 7 shows a full cloud CMTS deployment example on the AWS platform, where the UPF and RTC functions are placed at the edge and the nRTC, NRTC, and MP functions are placed in the Region.

The AWS virtual private cloud (VPC) service is used to form a virtual network to logically isolate the edge and Region resources for the CMTS. The CMTS VPC is connected to the on-premises network via an AWS Local Gateway, which further connects to the CIN and the SP core network via a SP router (SPR). This setup allows the DOCSIS upstream and downstream data traffic to traverse the same path as in the vCMTS today. For the CMTS control and management traffic, a dedicated connection known as AWS Direct Connect, is used between the Region and the edge to reduce the transport latency and jitter.

Within the Region, multiple AZs are used to add geo-redundancies for the control and management plane. The nRTC is placed on the resource-aware platform meet to the near-real time latency requirement. With explicit configurations, auto-scaling can be enabled for the nRTC across multiple AZs. The NRTC and MP can be placed on the serverless platform, thanks to the extra timing budget, which has built-in auto-scaling capability and better cost efficiency with its fine-grained, transaction-based cost model.

To support the user plane and control/management plane separation, the following interfaces/protocols are used to facilitate the communication between the CMTS service endpoints across the Edge and the Region. Each CMTS endpoint is assigned with an IP address from its hosting subnet reachable by other endpoints through the border gateway protocol (BGP).

Punt Path:	For the US UPF to punt the upstream DOCSIS signaling to RTC, nRTC or NRTC via TCP or UDP, such as CM Bandwidth Requests to RTC, Ranging Requests to nRTC, and Registration Requests to NRTC.
Inject Path:	For the CMTS control plane to inject the DOCSIS signaling messages to the DS UPF via TCP or UDP, such as MAP and UCD messages from RTC, Ranging Response messages from nRTC, and Registration Responses from NRTC.
Rest API:	For MP to configure/manage the CMTS control plane and user plane services. The REST APIs can be carried over HTTP and leverage existing interface definitions based on the DOCSIS YANG data models.
Telemetry Streaming	For MP to monitor the CMTS control plane and user plane operations. The telemetry data can be carried in GRPC and using the existing DOCSIS model driven telemetry interface definitions.

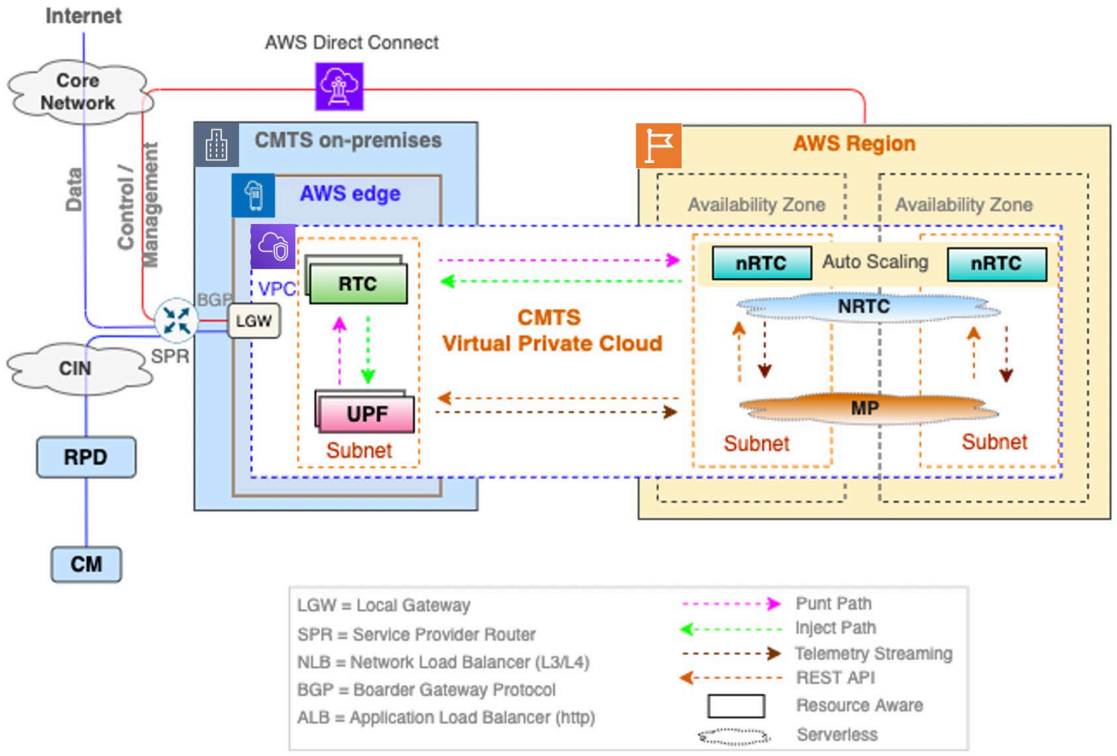

Figure 7 – Full Cloud CMTS Deployment Example on AWS

9. Serverless DOCSIS State Handling

Given the simplicity and economics appeal of the serverless platform, we are interested to explore how it can be used to support DOCSIS which is known for being deeply stateful. The DOCSIS control protocol is based on states of several DOCSIS entities, such as service groups (SGs), cable modems (CMs) and service flows (SFs). DOCSIS applications are multi-step procedures, such as profile management, load balancing and channel resource management for resiliency. The cloud serverless platform on the other hand has no affinity to the underlying compute infrastructure, and it is not possible to hold states across invocations. To provide serverless DOCSIS services, the code needs to be written in a stateless style.

Given the serverless technologies available today, the following two methods can be used for DOCSIS state handling,

- using the serverless data stores to persist the states externally, and
- using the serverless workflows to orchestrate the stateless functions.

This stateless approach to programming is also required to implement horizontal scalability which provide elasticity and thus better economics with resource-aware platforms.

9.1. Using the Serverless Data Stores

This is a key technique in the cloud native design that uses the external datastore to persist the states across function invocations. States are externalized as events passing between the functions through the data store while the functions themselves remain stateless. Examples of the datastores on a serverless platform include Amazon DynamoDB, an NoSQL database with single-digit millisecond access time or microsecond access time with cache accelerations, and Amazon S3 for large binary object storge.

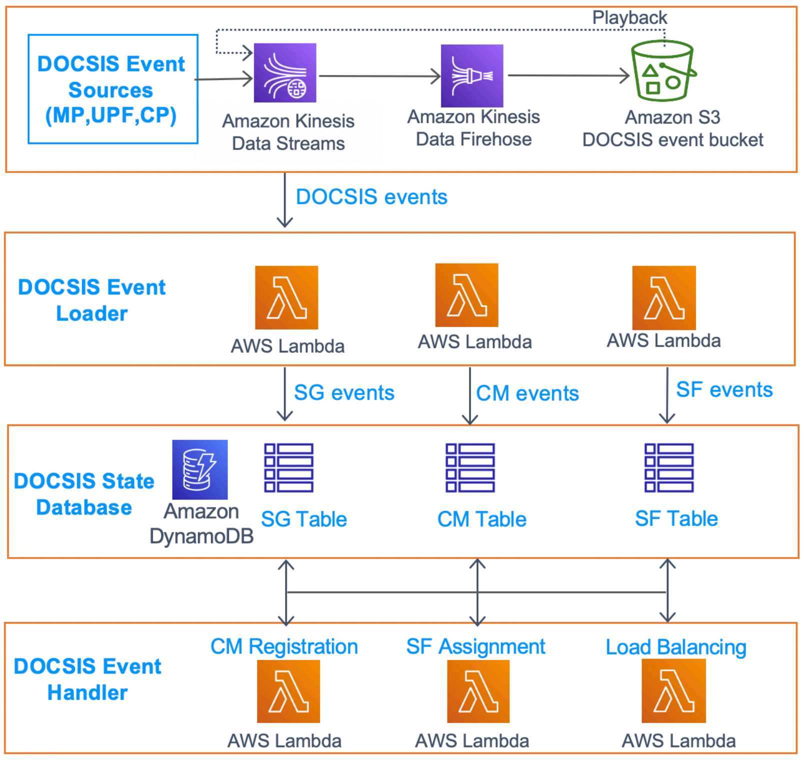

Figure 8 – DOCSIS State Handling Using Stateless Datastore

Figure 8 shows an example for using the serverless data stores to handle states of the non-RTC workloads. It uses an AWS serverless platform that contains AWS Lambda for serverless compute, Amazon DynamoDB for serverless database, Amazon Kinesis for serverless streaming, and Amazon S3 for serverless storage.

The design is based on an event driven architecture where the state information is passed as events between the stateless functions. The state changes of the DOCSIS SG, CM, and SF entities are represented as events from the management, user plane and control plane. These events are pushed into Amazon Kinesis, a data streaming service that keeps the event sequences in different shards or service groups.

These events are then pulled by the DOCSIS event loaders (implemented as AWS Lambda functions) and sorted into the corresponding Amazon DynamoDB tables. The loaded events will then trigger the DOCSIS Event Handler (also implemented as AWS Lambda functions) that watch the tables to process the state changes.

As an optimization, multiple handlers can be triggered for a single event. For example, a CM REG-REQ arrival event will trigger the CM Registration handler to process the REG-REQ, the Load Balancing handler to process the CM's load balancing group configuration, and the DS/US resiliency function to provide the channel list for the CM's transmit and receive channel sets.

In this example, there is also an Amazon S3 bucket to store the event history in longer term, which serves as the single source of truth that can be used to derive the DOCSIS states for debugging, fault recovery and AI/ML analytics.

In summary, by converting states into events, a stateful DOCSIS microservice can be disaggregated into a set of serverless functions. The fine-grained stateless functions then allow the serverless platform to quickly adapt the computational resource allocation to the shifting workload requirements, therefor achieving higher resource efficiency and lower cost comparing to the resource-aware platform.

9.2. Serverless Workflows

The DOCSIS application is typically a higher-level construct that contains various functions working in coordination with each other to accomplish the desired tasks. The execution order of the functions reflects the application states and the state transitions as the dependency or precedence of the execution of one function to another.

On the cloud serverless platform, there is a workflow service that can be used to record the function execution sequence, such that individual functions do not need to know about the application states. Additionally, since the workflow captures the prior knowledge of the execution order at runtime, it can be used for the platform level optimizations, for example, by warming up functions, pre-fetching data and instructions to achieve better performance.

Figure 9 is an example of using the AWS step function to implement DOCSIS profile management. AWS Step Functions is a serverless workflow service that can be integrated with other AWS services, such as AWS Lambda, Amazon DynamoDB, Amazon Kinesis, and Amazon SNS to accomplish a complicated procedure with many logical steps.

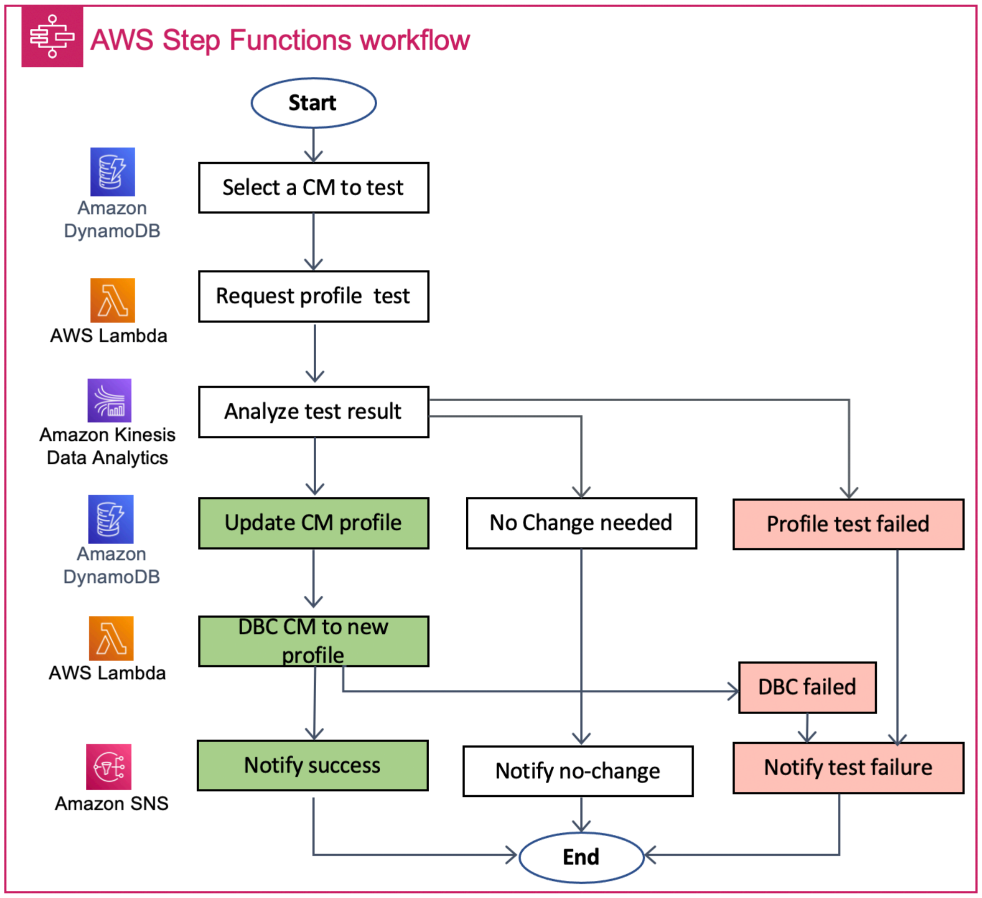

Figure 9 – DOCSIS Profile Managment Using Stateless Workflows

10. Getting Started

Moving to cloud is an interesting and challenging paradigm shift. It requires deep understanding of both the cloud services and the CMTS resource usage characteristics to maximize the cloud benefits. Since it may take years to mature to that level of understanding, cloudification is an iterative optimization process that requires learning while doing.

Given the on-prem CMTS has already been virtualized, a lift and shift strategy to the cloud is feasible to jump start the cloudification by moving a copy of an existing vCMTS and data to public cloud with minimal or no redesigning or modifications. This essentially is a test-driven development model to identify issues caused by the cloud hosting environment, and resource consumption hotspots for targeted redesign and optimizations.

This cloudification strategy motivated us to start a proof of concept (PoC) project using AWS as shown in Figure 10. In this setup, the vCMTS, the Cisco cnBR and the cnBR manager, are placed in an AWS Region that covers the on-premises lab location where the RPD and CMs are located. The cnBR in the Region is connected to the lab's router via an AWS Direct Connect. The PoC is used evaluate all vCMTS workloads in the cloud environment. Finding of this study will help refine the cloudification strategy and determine the architecture choice.

The in-region full cloud solution as used in this PoC can also be used as a convenient testing/development environment, which can be quickly created on-demand without the expansive lab expansions.

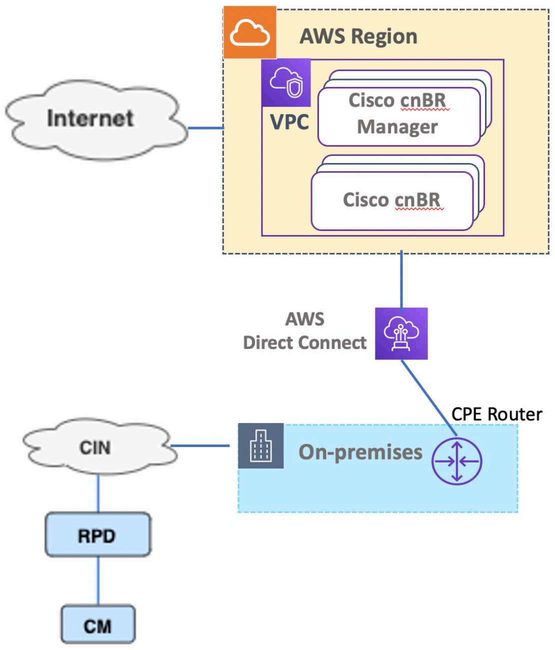

Figure 10 – Poof of Concept for CMTS Cloudification

11. CMTS-as-a-Service

With cloudification, the CMTS can be delivered as a service to the cable operators through the Internet or the direct connect services offered by the cloud providers. To make the CMTS-as-a-Service successful and sustainable, multi-tenancy is the key to bring down the CMTS development and maintenance cost.

In its basic definition, the CMTS multi-tenancy is an architecture in which a single CMTS entity serves multiple cable operators. Such entity can be the continuous integration and continuous delivery (CI/CD) pipeline that produces the production software and various cloud resources required to execute the CMTS functions.

11.1. CI/CD Simplification

CI/CD can be pictured as a pipeline, where new code is submitted on one end, tested over a series of stagers (code, build, test, release, deploy) and then published as operation-ready code. In the traditional single tenant environment, the complexity of the CI/CD pipeline goes linearly with the number of tenants. The CMTS vendor needs to maintain separate pipeline for each operator, slowing down the CI/CD life cycle, and wasting development /deployment resources.

With multi-tenancy, only one CI/CD pipeline is required as shown in. Figure 11, which can dramatically reduce the CI/CD complexity, improve quality and accelerate innovations, as all tenants would use the same code, the same operational infrastructure, and the sane deployment procedures.

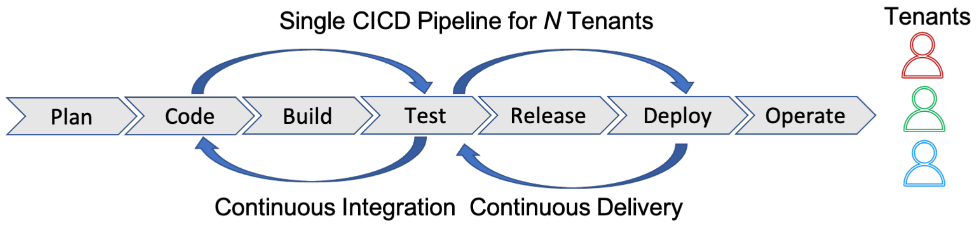

Figure 11 – CI/CD Simplification With Multi-tenancy

11.2. Multi-tenant CMTS Resource Sharing/Isolation

A multi-tenant CMTS achieves the cost and operational efficiency by sharing the same CMTS software and the cloud infrastructure among multiple operators, while keeping the tenants from accessing another tenant's resources. It boils down to a constrained optimization issue to properly partition the CMTS resources among the tenants either statically and/or dynamically.

There are a wide range of factors to consider for the resource partitioning across different tenants [6]. For example, a big operator who has large and constant workloads may be suitable for dedicated resources, a small operator on the other hand may find it more feasible to use the multi-tenant pooled resources. The workload characteristics also impact the partition strategy. For example, it's more economical to use the pooled resources for the DSX transactions, as they are only triggered occasionally. The goal of the multi-tenant CMTS is to efficiently serve everyone and automatically scale up and down based on current requirements demanded by the tenants.

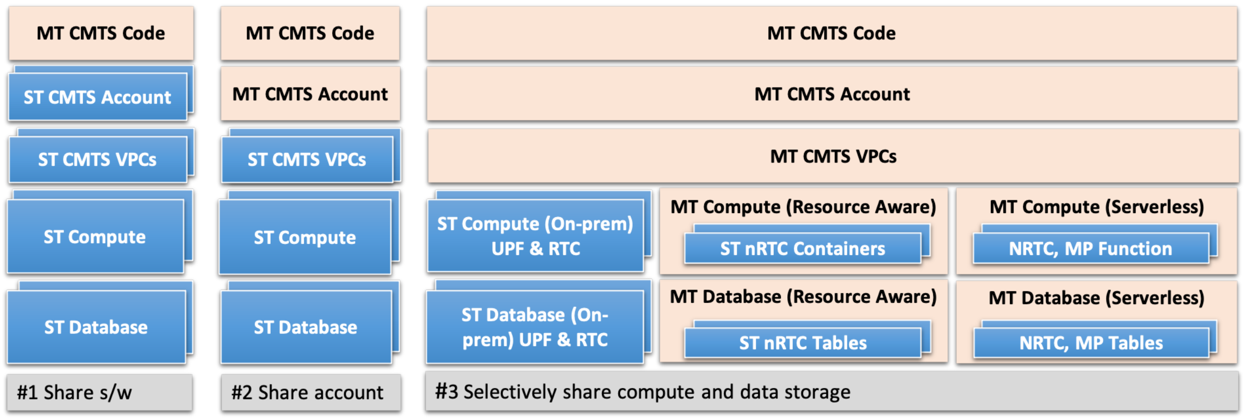

Figure 12 – Multi-tenant CMTS Resource Sharing and Isolation Options

Figure 12 shows three different resource partition options of the tenant stack including CMTS software, CMTS account, the VPC network construct, and the compute/data storage resources. These options may be applied in any combination to fit various tenant requirements.

The first option is the most isolated approach, where tenants only share the multi-tenant (MT) CMTS software. A single-tenant (ST) account is assigned to each tenant to keep all the moving parts of the tenant stack in complete isolation.

The second option pushes the isolation to the next level with all tenants running the same code in the CMTS provider's account. One or more unique ST CMTS VPCs are assigned to each tenant to enforce the isolation with the network boundaries. VPC based isolation avoids the certain account provisioning restrictions and gives the CMTS provider more centralized control across all tenants' deployment and operation. However, this option also has the same scaling issue as the first option, as the number of VPC grows, the management agility and resource efficiency decrease.

The third option has much more granular resource partitions among the tenants. It offers the best efficiency, agility and cost benefits among the three options. However, it requires a much more comprehensive tenant isolation strategy. Since operators are only naturally separated by their headend/hub locations, the on-premises UPF and RTC functions are completely isolated in compute infrastructure and data stores. For the CMTS services hosted in the Region, tenants are isolated by containers on the resource aware compute platform, and by functions on the serverless compute platform.

For the data stores, tenants can be isolated by tables or even table entries via the tenant identifier. Multiple cloud utilities are available to simplify tenant isolations. For example, using AWS Identity and Access Management (IAM) to implement run-time, policy-based isolations, using Amazon Elastic Kubernetes Service (Amazon EKS) namespace to isolate the ST container groups, and using the Amazon DynamoDB partition key to isolate ST tables.

12. Conclusions

The decision for moving the CMTS to the cloud is obvious: you pay for what you use, scale when you need, and spend less time managing the infrastructure/platform, and end up with more time to innovate for your business.

Cloud is different from the on-premises server environment with its distributed Region and edge locations and various resource-aware and serverless compute/storage platforms. Moving the CMTS to the cloud cannot be a simple rehosting process. Instead, it requires iterative optimizations to make most of the cloud environment.

With cloudification, the CMTS can be delivered as a service, enabled by CI/CD and the cloud based multi-tenant architecture. This opens new business opportunities for both CMTS vendors and cable operators improving business agility, productivity and cost efficiencies.

With the virtualization laying out the foundation, the cloudification of the CMTS can start with a lift-and-shift followed by a test-driven development model for targeted redesign and optimizations.

It is the time to unleash the power of the cloud computing for the CMTS.

Abbreviations

API	Application programming interface
AWS	Amazon Web Services
BGP	Boarder Gateway Protocol
CM	Cable modem
CCAP	Converged cable access platform
CIN	Converged interconnect network
CI/CD	Continuous integration, continuous delivery
CMTS	Cable modem termination system
CP	Control Plane
DS	Downstream
DSX	Dynamic service addition/change/deletion
FMA	Flexible MAC architecture
GCP	Generic control protocol
gRPC	Google RPC
MAP	DOCSIS bandwidth request
MP	Management Plane
MT	Multi-tenant
nRTC	Near-real time control plane
NRTC	Non-real time control plane
RPD	remote PHY device
RTC	real-time control plane
SG	Service Group
SF	Service Flow
SP	Service Provider
ST	Single tenant
UPF	User plan function
vCMTS	Virtualized CMTS
YANG	yet another next gen (data modeling language)

Bibliography & References

[1] https://www.o-ran.org/
[2] "CM-SP-R-PHY-I14-200323: DOCSIS Remote PHY Specification", CableLabs, 2020
[3] https://aws.amazon.com/about-aws/global-infrastructure/regions_az/
[4] https://aws.amazon.com/ec2/
[5] https://aws.amazon.com/serverless/
[6] https://aws.amazon.com/partners/programs/saas-factory/tenant-isolation/