Software-Defined Service Orchestration for MABR TV Services

A Technical Paper prepared for SCTE•ISBE by

Prabhu Navali
Engineering Manager
Ericsson Media Solutions
43 Nagog Park, Acton MA 01720
Prabhu.navali@ericsson.com

Raj Nair
VP, Technology
Ericsson Media Solutions
43 Nagog Park, Acton MA 01720
raj.nair@ericsson.com

Table of Contents

List of Figures

1. Introduction

Pay TV operators are increasingly moving their existing traditional on-prem only infrastructure-based Multicast-assisted ABR (MABR) TV services to a new next generation distributed deployment architecture – a software defined, media optimized container applications based microservices deployed across multi-cloud (operators public cloud, operators edge cloud and operators on-prem private cloud) infrastructure. This paper explores the composition of such a multi-cloud-based deployment architecture and details software-defined service orchestration of such a distributed platform.

2. Software-Defined MABR TV Services Orchestration

Traditionally, PayTV operators have deployed the managed MABR TV services from operators national and regional data-centers using on-prem components - for video encoding/transcoding, packaging, content protection, user experience, subscriber, multicast data plane and control plane management components, etc. However, recent trends in the industry and standards have made way for these components to be deployed in a multi-cloud environment.

2.1. SCTE Multicast ABR Reference Architecture

Increasingly operators are deploying PayTV platforms in a hybrid deployment model in which some components are deployed on-prem data centers and some components are deployed in public or private cloud. Recognizing this growing trend SCTE WG7 working group has been updating the reference architecture for delivering the MABR TV services.

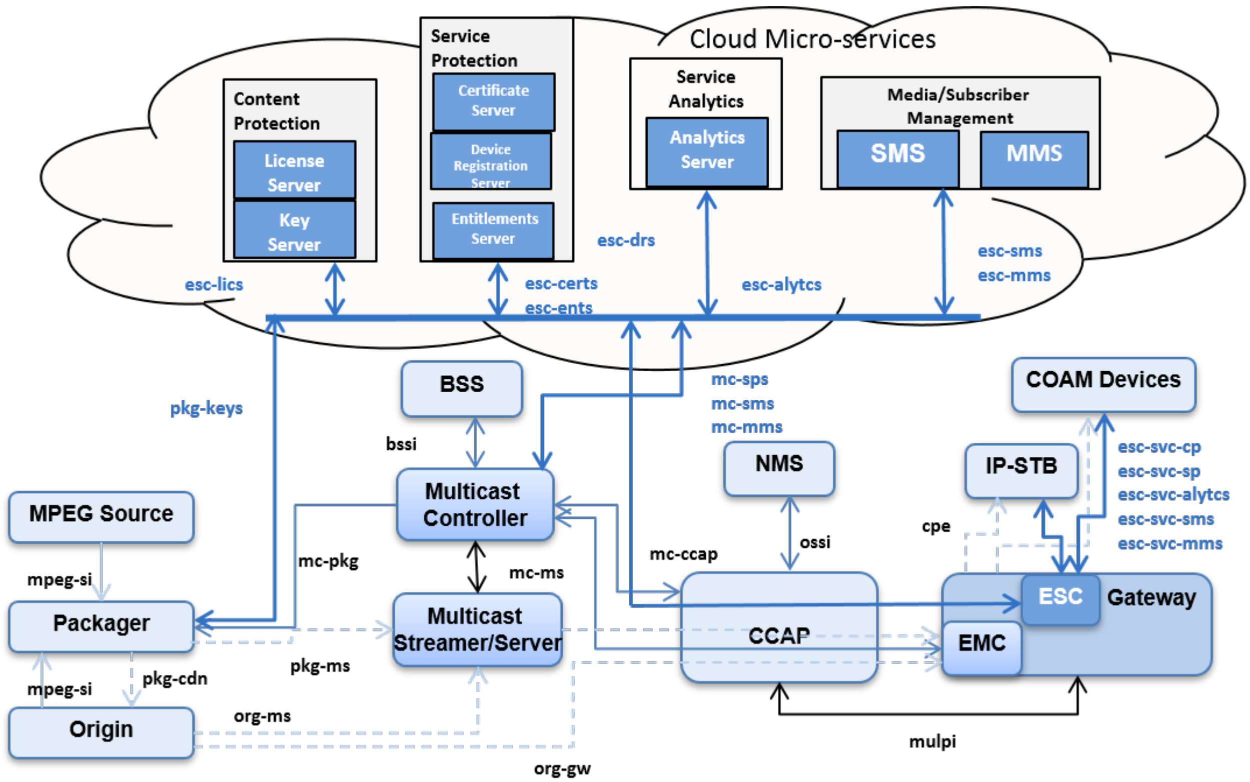

Figure 1 - SCTE Multicast ABR TV Services Reference Architecture

2.2. Drivers for the converged Multi-Cloud deployment of MABR TV Services

Rise of (operator) Public Cloud

Over the years with the rise of cloud computing, MABR TV services are being deployed in a hybrid deployment model - using (operator) public cloud and on-prem components in operator national/regional data-centers. (Operator) Public Cloud computing offers elastic compute/storage access allowing infinite elastic scaling, and utility-oriented usage. However, in order to scale the MABR TV Services workflows to millions of subscribers with good KPI metrics – there are challenges with the Hybrid deployment models. The real-time response and concurrency requirements are driving the cost of the cloud deployments. With ever increasing OTT clients (mobile and OTT STBs) there is a clear need to distribute the applications and workflows closer to the OTT clients on edge networks.

Rise of (Operator) Edge cloud

In order to address the latency and service scaling issues, operators are moving compute and storage resources to the edge network. The industry and standards are responding to this edge cloud evolution with development of architectures and standards [FOG, MEC].

Edge cloud extends the cloud computing to the edge network. Edge cloud facilitates the operation of computing, storage, and network services between OTT end points and MABR cloud services.

Rise of (Operator) On-Prem Private Cloud

In order to improve the efficiency of the compute, storage and network resources operators have been moving towards deploying private cloud infrastructures (like MaaS/OpenStack) in their national and regional data centers.

Rise of Containerized media optimized microservices applications

MABR TV Services are moving towards containerized microservice applications deployed over Container Clusters (like Kubernetes) over a cloud infrastructure.

Rise of DevOps Model for MABR TV Services deployment

Traditionally, for on-prem deployed MABR TV Services, features are developed and rolled out slowly. With the advent of DevOps model, the operators can roll out new features, patches more frequently. In addition, with cloud deployment scaling of the services can be done in short notice. The DevOps model benefits - TBD

The above changes are driving a paradigm shift in the way MABR TV services are being deployed - paving way for a new managed converged multi-cloud deployment of MABR TV Services.

2.3. Converged Multi-Cloud deployment of MABR TV Services

In the new paradigm MABR TV services are deployed over a converged multi-cloud deployment model.

The new deployment model includes MABR TV services deployed over multiple cloud infrastructures. This multi-cloud deployment can be vendor and operator co-owned and co-managed services model.

- Operator Public Cloud
    - Containerized microservices applications
- Operator On-Prem Private Cloud
    - Private cloud infrastructure in national and regional Data Centers
    - Containerized microservices applications
- Operator Edge Cloud
    - Private/public cloud in Edge Network (CO, Edge data centers, POP, MEC, etc.)
    - Containerized microservices applications
    - Intelligent edge caching/proxying microservice applications
- Multicast Core Network, Multicast Access Network, and Multicast in Edge Cloud
- Multicast-Assisted ABR – MABR (Multicast to Unicast conversion at the Edge).
- DevOps Model for MABR TV Services deployment.
- Microservices, Docker containers and Kubernetes clusters and orchestration
- Service discovery and scaling based on the requirements.
    - Service discovery and load-balancing frameworks
- ELK service monitoring and analytics services

2.4. Need for SDSO for MABR TV Services

The evolution of MABR TV Services deployment over a converged multi-cloud deployment model architecture needs microservices containers to be deployed across multiple public and private clouds.

The benefits of such a converged multi-cloud deployment are:

- Enormous scale
- Low latency response and improved KPIs
- Locality preserving
    - Local Ad insertion, personalization of TV services
- QOE management
    - Better user experience
- DevOps model
    - Speed of innovation and delivering new features/services to customers

The success of such a deployment depends on the ability to orchestrate these services across multiple public/private cloud infrastructures. The different clouds (public, on-prem private and edge cloud) may have different cloud infrastructures – for example, AWS, Azure, OpenStack, etc. Containerized applications allow one to deploy the same application across different cloud infrastructure. So, containerization normalizes the application deployment across different cloud infrastructure platforms. However, there is a good need for a software-defined service orchestration architecture to normalize and facilitate the orchestration MABR microservices.

2.5. Reference Model

2.5.1. Converged Multi-Cloud Deployment Reference Model

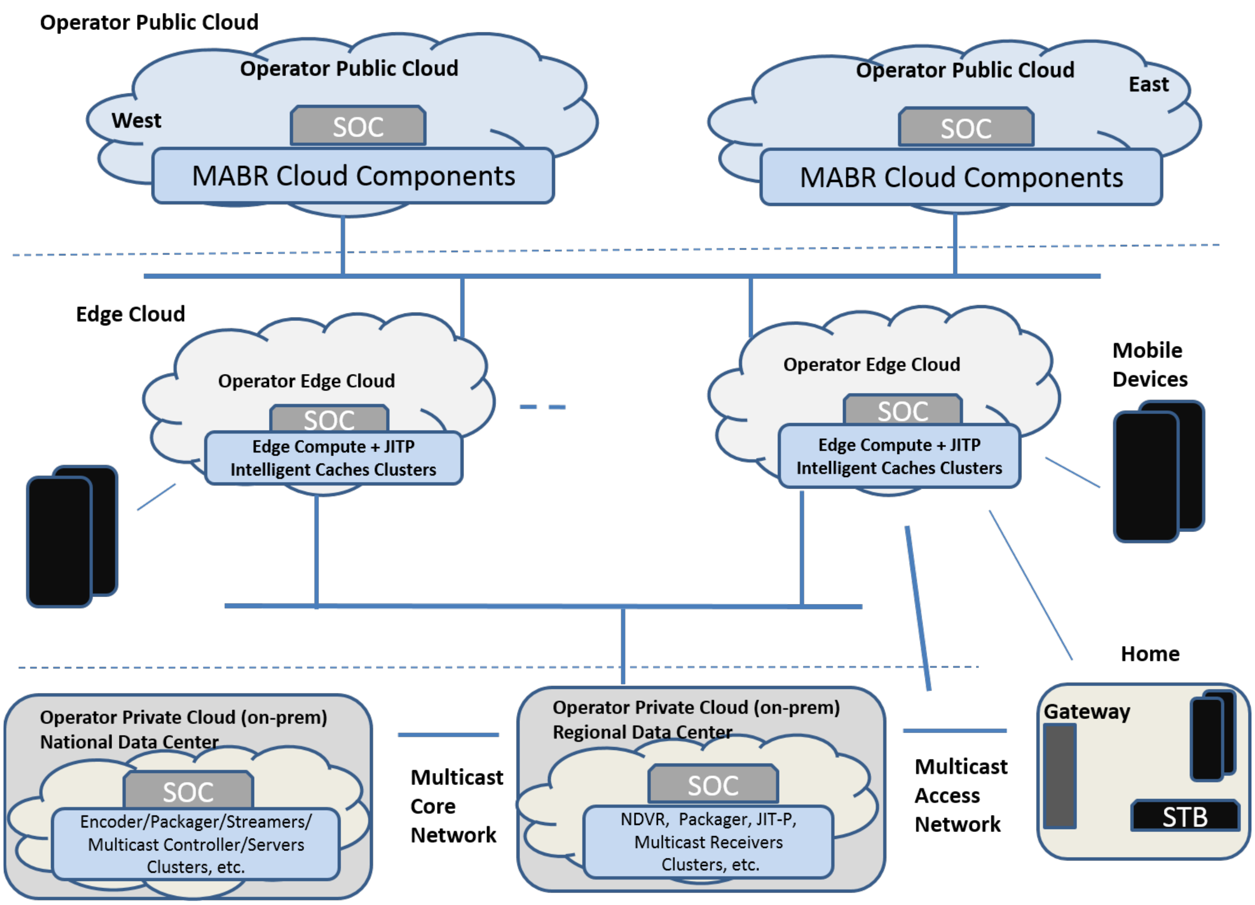

Figure 2 - MABR Services Multi-Cloud Deployment Reference Model

Figure 2 shows a reference deployment model for a distributed MABR TV Services. The MABR TV Services backend microservices components are distributed across public cloud, operator edge cloud and operator on-prem private cloud infrastructure. The services are made up of different layers or nodes that are distributed across different clouds.

The common denominator for these services is that this fabric made of SOCs that distributes the resources and services of computation, communication control, storage across all the available devices, systems, storage, clusters, cluster managers across the multi-cloud environment.

The network of software-defined *Service Orchestration Controllers* orchestrates and continuous management of multi-cloud MABR TV Services deployment.

2.6. Overview of Multi-Cloud MABR TV Services

Main functionalities delivered by the Multi-cloud MABR TV services are Live/Linear TV and Tim-Shifted TV services. Figure below shows an example high-level functional block diagram of microservices container clusters deployed across multi-cloud environment to deliver those TV services. Note that this is an extension of the SCTE MABR reference architecture adopted to multi-cloud deployment model.

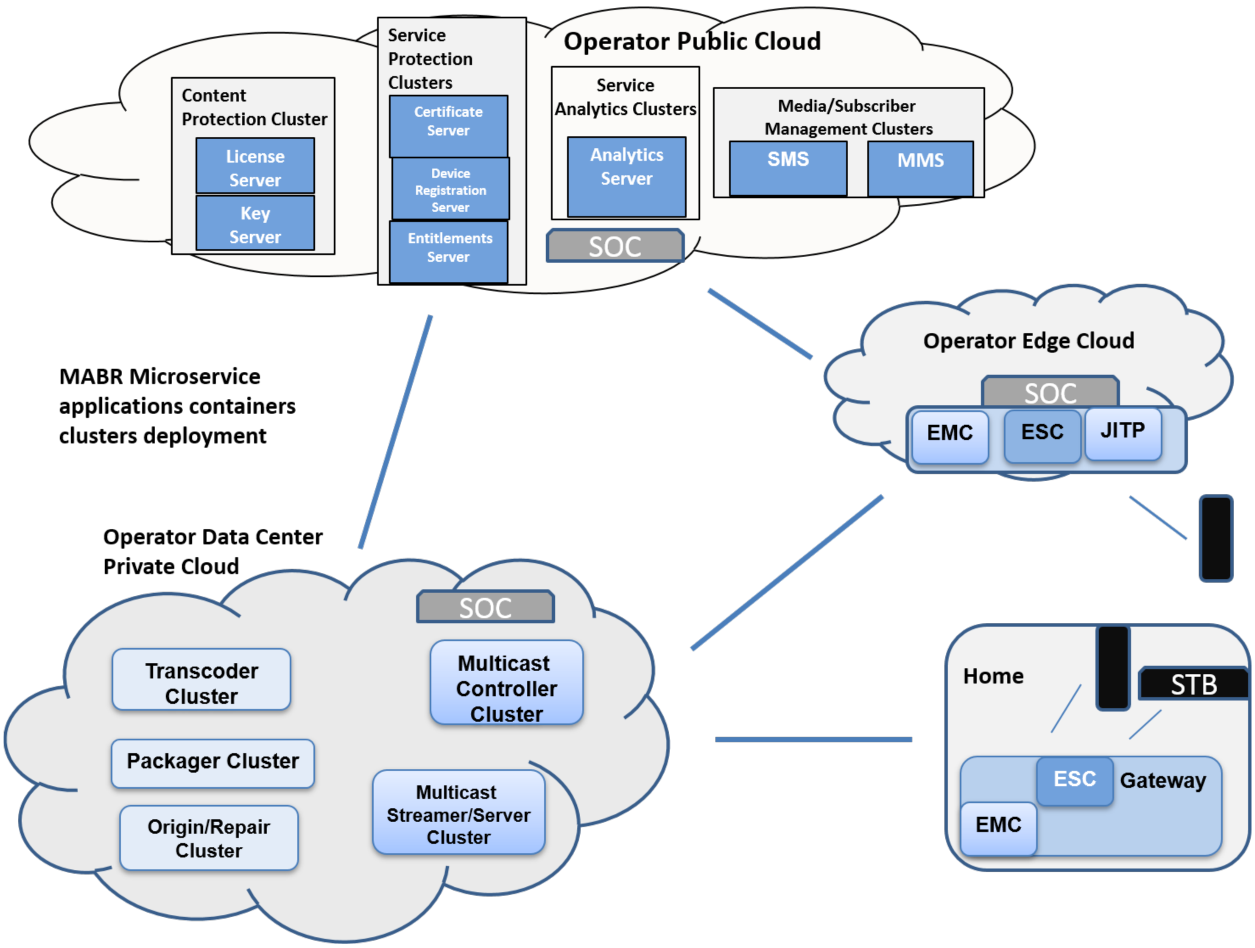

Figure 3 - Example Multi-Cloud MABR TV Services Deployment Components

- Channel Ingestion for Multicast ABR distribution

Involves in ingesting MPEG-TS streams and metadata into the Media Manager and Transcoder/Packager to generate Multi-bitrate multicast ABR MPEG-TS streams or ISOBMFF segments. The MABR streams or segments are encrypted with appropriate Common encryption schemes. The MABR streams are conditioned according to SCTE Adaptive Transport Stream (ATS) with Virtual Segmentation information for the downstream nodes.

- Multicast Data Plane
  There are two main multicast data plane options to distribute the ABR content over multicast.
  - Multicast ABR Streams Distribution

In this low-latency option, the ABR streams are sent over as MPEG-TS multicast streams to the downstream nodes like gateways or edge media routers (EMR)). In addition to carrying the a/v elementary streams, these transport stream may carry associated CMAF/ISOBMFF metadata (no Mdat) in a separate metadata elementary stream. The hybrid Stream+File format is called Common Mezzanine Distribution Format (CMZF). The residential gateway or EMR downstream performs a transformatting function on the received elementary streams to generate HLS or DASH segments for ABR delivery. The ABR segments are delivered using a local HTTP server to OTT clients. The associated manifest is fetched from the origin server. The MABR transport streams can be RTP encapsulated and can use existing IPTV RTP repair mechanisms for repair/recovery and faster channel tune-in.

- Multicast ABR segments Distribution
  In this option a file-based multicast protocol like NORM will be used to distribute the ABR segments over multicast. The Transcoder/Packager generates the ISOBMFF segments/manifests and are uploaded to origin server. Multicast server is used to distribute the ABR segments over NORM protocol. The downstream nodes like residential gateways with embedded multicast client (EMC) receive these segments over NORM and delivered using a local HTTP proxy cache to OTT clients. The origin/repair server will be used to perform repair and channel tune-in.

- Multicast Control Plane
  The multicast control plane includes multicast controller and its interactions with other nodes in the multicast data plane to control the delivery of channels over multicast. Some of the functions include – content selection, channel map management, reporting and bandwidth control functions.

- MABR Service Control Plane
  The service control plane includes backend components that are deployed in the public cloud and an intelligent cache or proxy or function in the residential gateway or in a node in edge cloud.

  This control plane provides many backend functions for the MABR TV services delivery – like subscriber management, user experience workflows, media life-cycle management, media databases, time-shifted services, playback control services, content protection services, service protection services, service discovery, load balancing, service access control, license servers, key servers, analytics servers, OSS/BSS integration, EPG ingestion/management, etc.

  The residential gateway or edge cloud node may include a embedded service control (ESC) component that includes intelligent caching and proxying functions for faster channel tune-in and to scale the MABR services. The caching and proxying component is used to cache/proxy for functions like - content protection, licenses, keys, service protection services, playback business rules (PBR), media and service entitlements, parental control.
  Most of the MABR service components are deployed as media optimized containerized microservice applications in on-prem private cloud, public cloud, operator edge cloud environments and residential gateway component.

2.6.1. Operator On-Prem Private Cloud Components

The operator On-Prem hosts a number of components, mainly focused on the content preparation, transcoding, packaging and multicast distribution. Most of the multicast data plane components are deployed on on-prem private cloud. Most of the components will be deployed as containerized microservices application clusters. Some of these components clusters can be realized either in edge cloud or public cloud as well.

Some of the example component clusters are:

1. Transcoding containers cluster
2. Packaging (JITP) containers cluster
3. Multicast Streamer containers cluster
4. Multicast Server containers cluster
5. Multicast repair server containers cluster
6. Faster Tune-In server containers cluster
7. Network DVR server containers cluster

2.6.2. Operator Public Cloud Components

A number of MABR Services components are deployed in the public cloud, mainly focused on the Control plane components like - user experience delivery components, Channel Catalog management components, Playback stream control components, Time-Shifted services workflows and data management components, subscriber management components, content protection (DRM) components, EPG ingest and integration components, OSS/BSS integration components, service protection components, and service discovery and delivery components.

Most of the multicast ABR services control plane components are deployed on the Public Cloud. Most of these components will be deployed as containerized microservices application clusters. Some of these components clusters can be realized either in edge cloud or on-prem private cloud as well.

2.6.3. Operator Edge Cloud Components

A number of MABR services components are deployed in the operator Edge Cloud mostly focused on the intelligent caching/proxying components for multicast data/control plane functions and caching components for service control plane functions. Edge cloud components participate in caching/proxying and deliver of ABR segments/manifests, caching/proxying and delivery of service control metadata like – channel map, channel data, metadata, subscriber/channel entitlements, licenses, keys, etc.

The intelligent caching/proxying components include:

- Embedded Multicast Client (EMC) functions
    - Join/Leave Multicast groups to receive multicast data
    - Receive Multicast streams or segments
    - Multicast to Unicast conversion and preparation of Unicast segments for delivery
    - HLS or DASH delivery of manifests/segments
    - Channel Map managements
- Embedded Service Control (ESC) functions
    - Scaling MABR backend Services
    - Faster Tune-In
        - Enable faster channel playback to reduce playback latency - for channel tune-in, for Channel surfing, for channel switching
    - Intelligent caches enable Faster Tune-In by caching the relevant information needed for a channel playback

        - Caches information for one or more channels in the Channel Map
        - Caches the following to enable Faster Tune-In
        - Licenses (keys), program entitlements, entitlements, ratings, channel roll responses
    - Service protection

Some of the example component clusters are:

1. EMC multicast receiver containers cluster
2. JITP Edge Packaging containers cluster
3. ESC service control containers cluster
4. Stream/Service personalization/localization server containers cluster
5. EMC/ESC reverse proxy service containers servers

2.7. Unified Service Orchestration Infrastructure

As mentioned in the above sections for a successful delivery of MABR services – microservice applications are deployed as container clusters across multiple cloud infrastructures. The cloud infrastructure and microservice infrastructure services are leveraged to deploy the application clusters.

Cloud infrastructure services

The public and private cloud infrastructure provides services like IaaS/PaaS to orchestrate compute/VM, storage and network resources in a cloud environment. Example cloud infrastructure services are AWS, Azure Cloud, MaaS, OpenStack, VMware ESXi, etc.

Microservice application containers

Container software provides a standardized (normalized) way of developing, deploying microservice applications. Containers can be lightweight and share the host OS resources. Containers can be more portable by packaging all the required packages and OS resources into the container. Container are secure as the container engines provides resource isolation and access/resource security for the containers Containers typically run on a container engine (like Docker Engine) on a host (VM or native compute). The containers are portable and can be run on different cloud infrastructures.

Container cluster infrastructure services

Microservice application container clusters are deployed over the cloud infrastructures using container application cluster management and control platforms like OpenShift, Kubernetes, Docker Swarm, etc. These platforms support various types of container software – like docker, Mesos, etc.

Service discovery

Containers and container cluster managers like Kubernetes work with *Service Discovery* frameworks like (Eureka, Consul, Vault etc.). These frameworks focus on providing service registration and discovery services. Containers and cluster managers integrate with *Service Discovery* frameworks to register the containers microservice with related metadata. A *Service Registry* allows a requester (another container application or external applications) to discover the service hosted on different container clusters. The service selection from the service registry is based on the application requirements and constraints. There could be multiple service registries in a cloud, or one each in every cloud deployment, or a just one global *Service Registry* for the entire MABR services.

Service Access

Service Registry will be used to provide load-balancing across different microservice container clusters. A *Service Load Balancer* (like AWS ELB, NGINX, etc.) will request the available services (frequently updated/fetched) and will provide, update the service access features.

2.8. Software-Defined Service Orchestration

This section (and article) proposes a distributed *Software-Defined Service Orchestration* framework that distributes the MABR microservices throughout the multi-cloud environments. There is a clear need for such a framework given the nature of the microservices configuration, deployment constraints and different cloud infrastructure and microservices environments. The proposed framework allows such a distributed platform to perform orchestration of microservices using available nodes, containers, container clusters, cluster managers, service discovery platforms, storage, and networking resources. The distributed platform is a network of *Service Orchestration Controllers* that facilitates the orchestration tasks. The Distributed *Service Orchestration Controllers (DSOCs)* glue together the service container clusters that are running across multiple clouds (Public Cloud, Edge Cloud, Operator Private Cloud).

Microservice applications specify in the form of *Directives* high level tasks, configuration and requirements for the successful deployment and operation of the service.

These *Directives* are carried in *Service Manifests.* A *Service Manifest* is a document that carries the instructions on how to deploy, delivery, manage, operate a service. The *Service Manifest* is an XML/YAML/JSON based document and contains various *Directives.*

Some of the examples of *Directives* are:

1. Service application configuration data
2. Service graph/chart for deployment
3. Service constraints - CPUs, Bandwidth, Storage, etc.
4. Service cluster configuration data
    a. Auto scaling metrics, constraints, load-balancing constraints
5. Service access related data
6. Service delivery related data
7. Service monitoring, analytics related data

Service Orchestration Controller (SOC)

A distributed service orchestration platform is a network of *Service Orchestration Controllers (SOCs)* deployed in different cloud platforms. These *SOCs* distribute *Service Manifests* and perform the functions based on the *Directives* defined in the manifests. The global topology configuration of controllers allows SOCs to discover each other. Typically, each *SOC* is paired up with a local or global *Service Discovery* service through which it discovers and monitors and facilitates orchestration of application containers via container cluster managers (like Kubernetes). The distributed *SOCs* manage the life-cycle of the microservices applications containers. The *SOC* incudes a *Service Manifest Controller* which handles functions related to creating, updating, distributing, and receiving *Service Manifests.*

Some of the main functions of *SOC* are:

- Handle submission of a High-Level Service orchestration *Task*
  - Submission of a high-level request triggers the preparation of *Service Manifest* with *Directives* to fulfill the orchestration request
  - Service Discovery
    - This typically involves in the discovery of the service components across multi-cloud environment and determining the existing service components, constraints, infrastructure components, clusters etc. available to make adjustments to *Service Manifest Directives* to fulfill the new orchestration request.
- *Service Manifest Controller* functions
  - creating, updating, distributing, and receiving *Service Manifests*
- Handling Service *Directives*
  - Decomposition of *Directives* into smaller *Orchestration Tasks*
- Communicating *Orchestration Tasks* to Orchestration Managers (like Kubernetes)
- Communicating Orchestration request status, monitoring, etc.

The distributed *SOCs* handle several *Task*s to setup, teardown or update the microservices containers based on the MABR service business needs. A high-level service orchestration request example like - Live/Linear channels catalog ingestion - will trigger an *Orchestration Task,* with a customized *Service Manifest* with the appropriate *Directives,* to fulfill the request will be created at the public cloud *SOC* and distributed to other *SOCs* in the multi-cloud deployment. The *SOCs* in each cloud environment work with the Cluster Managers to fulfill the *Task* request – which includes service discovery, resource allocation, generating local orchestration charts/manifests to orchestrate the containers in different microservice clusters. The *SOC*s send updates on the progress of the *Task*. The distributed *SOCs* continue work together to handle any *Task* requests for setting up or tearing down or updating the microservices containers deployed across multi-cloud environment.

Conclusion

In this paper we prose a framework for a distributed *Software-Defined Service Orchestration* framework for the orchestration of MABR microservices applications across multi-cloud environments over disparate cloud platforms and normalized container cluster platforms. The MABR Microservices are deployed across Public Cloud, Operator Edge Cloud and Operator Private Cloud infrastructures. The distributed *Software-Defined Service Orchestration* framework provides mechanisms to distribute high-level orchestration requests (*Tasks*) into actionable *Directives* using *Service Manifests* upon which the local, global Cluster Managers, service discovery components and *SOCs* can act on. The *SOCs* in conjunction with local Cluster Mangers orchestrate the application microservice containers and manage the life cycle of microservices application containers.

There are industry trends in both moving the microservice applications to Edge Cloud for a multi-cloud deployment model, and software-defined orchestration frameworks to facilitate such a distributed application deployment.

We think there is a need for standardization of such a distributed software-defined MABR microservices orchestration framework.

Abbreviations

ABR	Adaptive bit rate
MABR	Multicast Assisted ABR
NORM	Nack-Oriented Reliable Multicast
SCTE	Society of Cable Telecommunications Engineers
SOC	Service Orchestration Controller
SDSO	Software-Defined Service Orchestration

Bibliography & References

Cablelabs Multicast-assisted ABR Technical Report.

SCTE Multicast ABR Services part -1: Data Plane Workflows Specification – SCTE DVS WG7 (under progress)

SCTE Multicast ABR Services part -1: Service Control Workflows Specification – SCTE DVS WG7 (under progress)

OpenFog Architecture Overview – White Paper, OpenFog Consortium Working Group

ISO/IEC 23009-1 - DASH Specification

RFC 8216 HTTP Live Streaming (HLS) Specification

ISO/IEC 13818-1 MPEG-2 Transport Streams Specification

ISO/IEC 13818-1 Edition 7 PDAM 1 – Carriage of associated CMAF boxes for audio-visual elementary streams in MPEG-2 TS. (under progress)

