Network Working Group
Internet-Draft
Intended status: Informational
Expires: 16 September 2026

R. Ramdhany
BBC
N. Race
D. King
Lancaster University
15 March 2026

Use Case and Challenges for the Deployment of Object-Based Media across the Internet
draft-rrk-object-based-media-usecase-01

Abstract

This document outlines the challenges and use cases for the deployment and operation of Object-Based Media (OBM), also known as Flexible Media (FM), across the Internet. It discusses key considerations such as compute-aware traffic steering, metric usage, bandwidth optimization, and latency reduction techniques.

The intention of this document is to highlight specific challenges or areas where IETF investigation and applicable solutions are needed for the optimal deployment of OBM-based media services.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://example.com/LATEST. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-rrk-object-based-media-usecase/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:WG@example.com), which is archived at https://example.com/WG.

Source for this draft and an issue tracker can be found at https://github.com/USER/REPO.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 16 September 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Object-Based Media (OBM) represents a significant evolution in media distribution, allowing content to be dynamically assembled, personalized, and optimized for different devices, user preferences, and network conditions. Unlike traditional linear media, OBM decomposes content into discrete objects such as video segments, audio layers, and metadata that can be orchestrated in real-time. This approach enhances adaptability, interactivity, and efficient resource utilization across media delivery ecosystems.

The deployment of OBM-based services introduces several technical challenges, particularly in terms of network infrastructure, compute resource management, and traffic steering. The integration of compute-aware networking techniques, such as those being developed by the IETF Compute-Aware Traffic Steering (CATS) working group, is crucial to optimizing OBM workflows. By leveraging standardized frameworks and emerging IETF technologies, OBM can be effectively deployed across diverse media ecosystems.

The intention of this document is to highlight open gaps or areas where IETF efforts are needed for the ongoing deployment of OBM-based media services. It outlines key OBM service types and their respective functionalities, emphasizing their implications for network and compute resource management.

1.1. Object-Based Media (OBM)

Traditional media broadcasts, in both radio and video, broadcast a packaged, edited, single linear stream of information to all users regardless of playback device or environmental factors. Object-based media represents a significant shift from traditional media production and broadcasting methods, focusing instead on creating, storing, and transmitting media as a collection of discrete objects (such as audio, video, or text elements) rather than a single, unchangeable stream. This approach allows for media to be more

interactive, adaptable, and personalised to individual viewers or listening environments, offering several advantages for the future of television and other media experiences.

Object-based (OBM) media, also known as Flexible Media (FM), enables content to be tailored to individual preferences or requirements. For instance, a viewer could adjust the level of background music in a program, switch between different camera angles, or select which storyline to follow in a complex narrative. This level of personalisation enhances viewer engagement and satisfaction. It can greatly improve accessibility features for diverse audiences. For example, audio descriptions for the visually impaired or sign language for the deaf can be seamlessly integrated and toggled on or off according to the viewer’s needs. This inclusivity expands the reach of the content to a wider audience.

Efficient use of Internet bandwidth, especially at scale, is crucial. Broadcasters can optimize bandwidth usage more efficiently by transmitting only the objects necessary for a particular viewer’s experience. This is particularly beneficial in environments with constrained bandwidth or users with limited data plans. FM media can be designed to be compatible across various devices and screen sizes, ensuring a consistent user experience whether the content is viewed on a smartphone, tablet, or large television screen. This scalability is crucial to the variety of devices and screens.

FM content can be stored on Content Delivery Networks (CDNs), which efficiently distribute digital content, such as multimedia files and live streams, over IP networks to multiple endpoints and viewers. Typically, a CDN includes one or more servers responsible for delivering digital objects or streams. Additionally, it features a management or control system that handles various operations such as content distribution, request routing, reporting, metadata management, and other functionalities essential for the system’s performance.

Contemporary approaches to media delivery are challenging for OBM content distribution. The flexibility in personalised media introduces a challenge of re-versioning, where the combinatorial explosion of potential content versions makes it impractical to pre-render and store all permutations. On-demand re-versioning can occur on the device if local processing capabilities and network bandwidth allow the objects to be transported to the device and composited. However, media object composition will likely need to be offloaded upstream in the distribution pipeline to enable universal delivery.

1.2. Significance of Object-Based Media

An OBM service is personalised media such as stories, audiobooks, and games. This content is generated based on user preferences and experiential learning, ensuring a tailored experience. It employs multi-directional delivery methods, allowing user-generated content and personalised media to thrive. This "software-powered content" caters to end-users, the primary recipients and participants in this dynamic ecosystem.

From a network distribution perspective, OBM media requires a robust and flexible delivery infrastructure capable of handling the dynamic assembly of content based on user interactions and preferences. This necessitates advanced content delivery networks (CDNs) and edge computing solutions that efficiently process and deliver personalised content streams. Moreover, the scalability of this approach is critical, as it must support a potentially vast number of unique user experiences generated from the same set of media objects. By separating media components, creators can offer multiple versions of content tailored to different needs, such as alternative audio tracks for different languages or visually impaired audiences requiring descriptive audio. This level of adaptability not only enhances the user experience but also broadens the audience’s reach.

Personalisation of media can occur in multiple stages if regional and personal preferences are considered. Delivery infrastructures for personalised media need therefore offer flexibility in creating dynamic media composition stages at various locations in the network to efficiently support the various personalisation permutations. Further, the selection of media composition sites depends on the availability of compute resources, proximity to storage for media objects and if user agency is afforded, round-trip times to the destination Client.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms are used in this document:

* AI: Artificial Intelligence aims to create systems capable of performing tasks that typically require human intelligence, such as understanding natural language, recognizing patterns, and making decisions.

* AI Agent: An autonomous software entity that perceives its environment, reasons over goals, and takes actions (including communicating with other agents) to fulfil a delegated task. In the OBM context, AI agents implement functional components such as the Flex Media Orchestrator, the Compute and Object Allocator, and the Network Orchestrator.

* Agent-to-Agent (A2A) Communication: A structured interaction model in which AI agents exchange messages to delegate tasks, advertise capabilities, report status, and negotiate resource usage. A2A communication requires a protocol capable of supporting these interaction patterns in a secure and interoperable manner.

* Clients: Media playback applications are designed to request and ingest flexible media content.

* Edge Computing: Is a computing pattern that moves computing infrastructures, i.e, servers, away from centralized data centers and instead places it close to the end users for low latency communication.

* Network Edge: Is an architectural demarcation point used to identify physical locations where the corporate network connects to third-party networks.

* Objects: Assets that are used to make a piece of content.

* Scheduler: Instantiates executors for jobs at a local controller based on the resources available at the compute site where the system resides.

* Service: A media stream of user-generated personalised content.

* Service identifier: An identifier representing a service, which the clients use to access it.

3. Deploying Object-Based Media Services

It is important to optimise network traffic in OBM environments where computing resources are distributed across multiple locations. This involves making the network aware of the computational context: where computing resources are located, their capabilities, and their current load or availability. By analyzing these factors, the network can optimize routing decisions, improving efficiency, reducing latency, and enhancing the performance of networked applications and services. Unlike traditional media formats, OBM treats media elements as independent objects that can be orchestrated at runtime. This document outlines the use cases and requirements

for OBM, emphasising the possible role of the IETF Compute-Aware Traffic Steering (CATS) initiative in optimising the delivery of OBM content. Key components of OBM include:

* Media Objects: Discrete elements such as video segments, audio layers, and metadata.

* Orchestration Engine: Determines how objects are assembled based on contextual factors.

* Delivery Mechanisms: Network protocols and architectures enabling object transport and synchronisation.

The dynamic nature of OBM necessitates advanced traffic steering mechanisms that can adapt to compute and network constraints in real-time.

3.1. Compute Aware Traffic Steering

Deployment of OBM services introduces several technical challenges, particularly in the context of Compute Aware Traffic Steering (CATS). Real-time adaptation of compute resource usage is a major challenge, as media objects must be dynamically composed and delivered based on changing network and compute conditions. Synchronisation across distributed compute nodes is also essential, ensuring coordinated media object delivery and processing across edge, cloud, and on-premise compute resources. As the process is managed an additional challenge of Quality of Experience (QoE) management, where compute resource usage must be balanced with perceived quality improvements to enhance the user experience, is also required.

To support OBM delivery, compute-aware traffic steering must fulfil several requirements. It must possess dynamic compute resource awareness, allowing assessment and adaptation to available compute power along the delivery path. Multi-layer orchestration is necessary to coordinate network-layer traffic steering with application-layer OBM composition. Low-latency compute routing is crucial for minimising processing delays in interactive media experiences. Additionally, scalability and load balancing are needed to ensure efficient distribution of media object processing, preventing compute bottlenecks. Lastly, edge-aware optimisation should be integrated to leverage edge computing for latency-sensitive OBM applications.

3.2. On Path Computing

The Computing-Aware Traffic Steering (CATS) framework is designed to enhance traffic steering by considering both network and computing resource metrics. The approach aims to optimize service delivery in environments where computing resources are distributed across multiple edge sites.

There are several components in CATS Framework:

* Service Sites and Instances: These are locations hosting one or more service instances capable of processing client requests.

* CATS Service Metric Agent (C-SMA): This component collects metrics related to the performance and availability of service instances.

* CATS Network Metric Agent (C-NMA): Responsible for gathering network-related metrics, such as latency and congestion status.

* CATS Path Selector (C-PS): Utilizes metrics from both C-SMA and C-NMA to make informed traffic steering decisions, ensuring optimal service delivery.

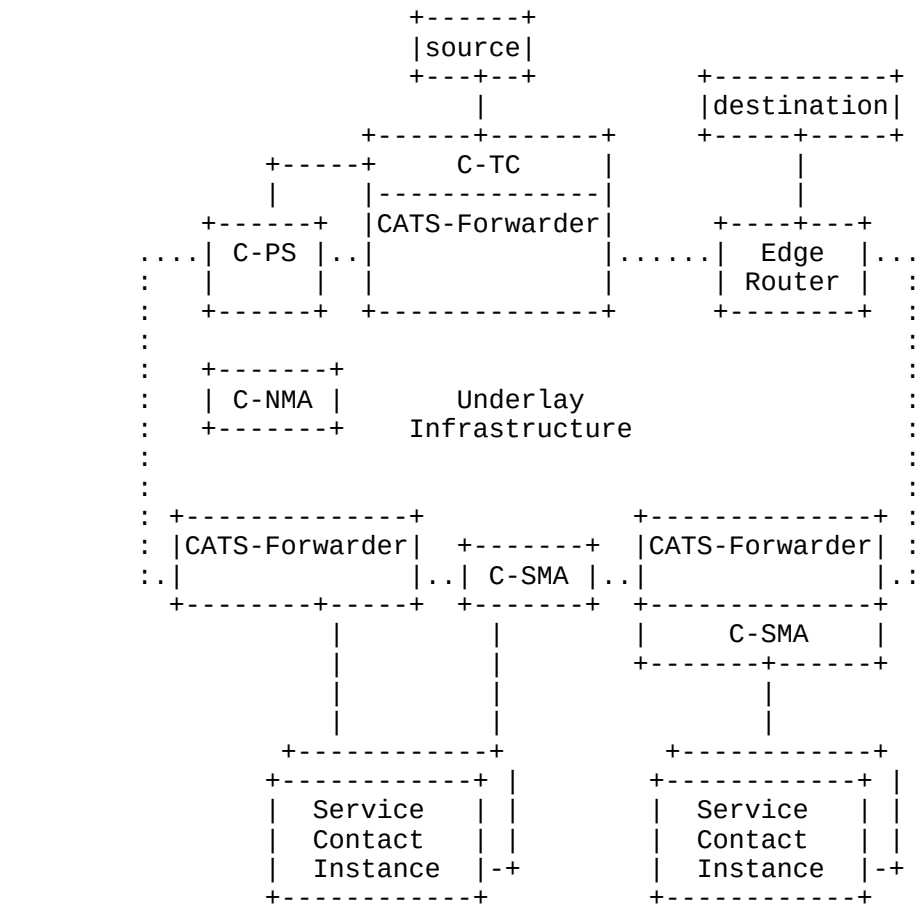
* CATS Traffic Classifier (C-TC): Identifies and classifies incoming traffic to apply appropriate steering policies.

The described workflow in the CATS framework assumes a classic client-server interaction, where a client submits a compute request, a server processes it, and the result is returned to the original requester. This model aligns with the current CATS architecture, in which the CATS Path Selector (C-PS) determines the most suitable compute instance and steers traffic accordingly.

There is an additional requirement where multiple compute instances "on the path" that the C-PS might choose and steer traffic. For example, a flex media experience will be steered between multiple compute service instances (media object compilers), before finally being steered to a CDN server to be cached, or back to user. These scenario would require a slight modification to the CATS framework, and is presented below.

The CATS framework describes a single compute instance performing the necessary computation before returning the result. Enabling on-path compute would require support for sequential steering, allowing C-PS to dynamically route traffic between multiple compute instances before delivering the final output. The CATS Service Metric Agents (C-SMA) and CATS Network Metric Agents (C-NMA) may also require enhancements to track multiple processing stages, ensuring each

instance is selected efficiently based on compute and network conditions. This extension transforms the CATS framework from a single-step compute selection model into a multi-hop, compute-aware path.



To fully realise OBM services, a new requirement exists where multiple compute instances may exist along the network path, and the C-PS may need to steer traffic between them sequentially rather than directly selecting a single endpoint.

For example, in a OBM experience, a request might be processed by multiple compute instances (such as a compute-heavy media object compiler and an additional GPU node for rendering 3D overlay graphics or providing descriptive audio narration) before finally being directed to a CDN server for caching, or back to the user that requested the service.

This approach introduces a multi-stage compute workflow instead of the single-step selection model in the existing CATS framework.

3.3. Bandwidth Optimisation Strategies

To be discussed in future versions of this document.

3.4. Latency Reduction Techniques

To be discussed in future versions of this document.

3.5. Media Transport Protocols

The Internet Engineering Task Force (IETF) has been involved in numerous initiatives and has developed various standards and protocols to improve and facilitate video media delivery across Internet infrastructure.

* Real-Time Streaming Protocol (RTSP): Defined in [RFC7826], RTSP controls streaming media servers. RTSP is designed to control media sessions between endpoints and acts as a network remote control for multimedia servers.

* Real-time Transport Protocol (RTP): Specified in [RFC3550], RTP delivers audio and video over IP networks. It is widely used in streaming media systems, video conferencing, and push-to-talk features (VoIP).

* RTP Control Protocol (RTCP): Defined alongside RTP in [RFC3550], RTCP provides out-of-band statistics and control information for an RTP flow. It monitors transmission statistics and quality of service (QoS) and aids in synchronisation between different streams.

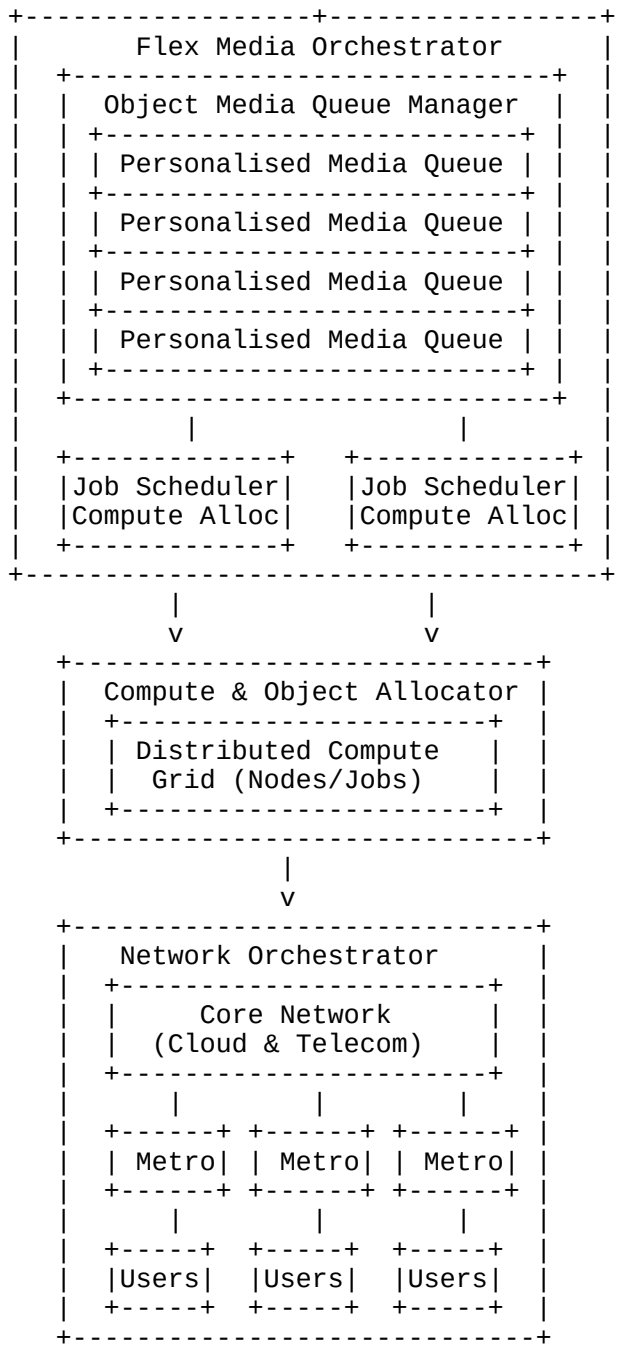
* HTTP Live Streaming (HLS): While initially developed by Apple and not originally an IETF standard, HLS has become widely adopted for online streaming live and on-demand video content. The IETF has documents that discuss HLS within the context of internet infrastructure, such as [RFC8216].

* Media Over QUIC (MoQ): To be discussed.

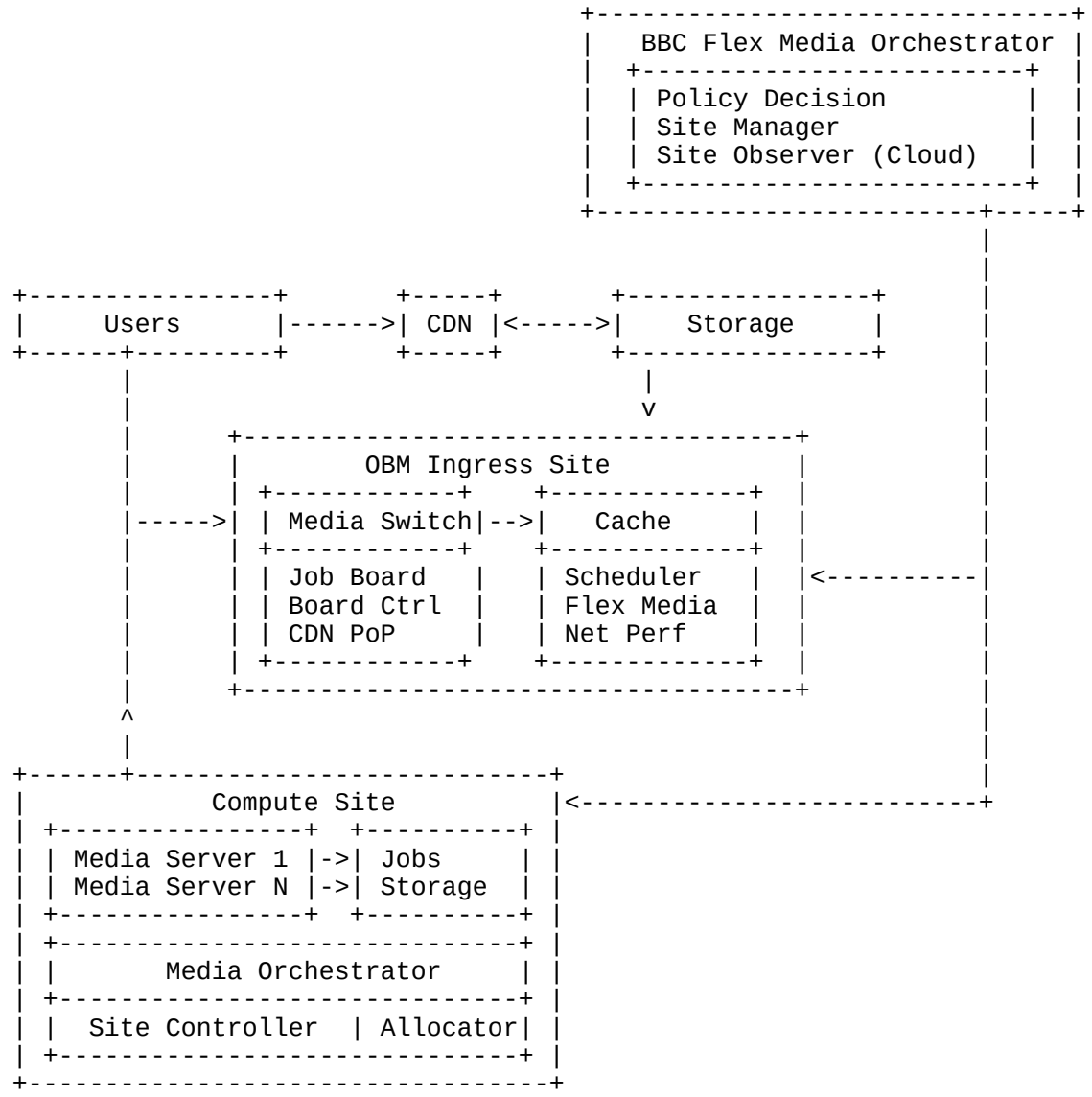
Future versions of this documented will highlight specific techniques that would bring benefits to the consumer (user) and provider of OBM services.

4. Architecture for Object-Based Media

The following sections discusses a high-level architecture and functional components for the deployment of OBM services.


Flex Media Orchestrator
Object Media Queue Manager
Personalised Media Queue
Personalised Media Queue
Personalised Media Queue
Personalised Media Queue
Job Scheduler
Compute Alloc
Job Scheduler
Compute Alloc
Compute & Object Allocator
Distributed Compute
Grid (Nodes/Jobs)
Network Orchestrator
Core Network
(Cloud & Telecom)
Metro
Metro
Metro
Users
Users
Users

The above figure provides a conceptualized architecture for flex media processing and delivery, where media objects are dynamically assembled and personalized for individual users. Unlike traditional media streaming, which delivers a pre-encoded linear stream, OBM decomposes content into discrete media objects such as video segments, audio tracks, subtitles, and metadata. These objects are retrieved, processed, and compiled dynamically based on user preferences, user device capabilities (or distributed and dedicated compute nodes), and network conditions.



Users are shown as interacting with a system that manages personalized media queues. Each user is associated with a dedicated queue that maintains a sequence of media objects tailored to their specific requirements. This approach allows for fine-grained control over content delivery, ensuring that media elements are customized in real-time. The Object Media Queue Manager is responsible for handling these personalized queues and forwarding processing requests to the appropriate compute resources. These components coordinate

the retrieval and processing of media objects by distributing tasks across available resources. The Job Scheduler determines the order and priority of processing tasks, while the Resource Allocator assigns computational and storage resources to execute them. This scheduling mechanism is critical for balancing resource utilization, optimizing media rendering, and ensuring low-latency delivery.

The Compute and Object Resource layer, which includes distributed compute nodes and storage elements responsible for processing media objects. The connections between the Job Scheduler, Resource Allocator, and compute nodes allow workloads to mapped to specific resources. The dynamic scheduling approach where different media objects are processed in parallel across multiple compute units, enhances scalability and efficiency, enabling personalized media delivery at scale. By integrating compute-aware resource scheduling with object-based media workflows, this architecture supports adaptive content distribution while optimizing network and compute resources.

4.1. AI Agent-Based OBM Architecture

The functional components described in the OBM architecture are well-suited to implementation as AI agent entities that autonomously reason, plan, and act to fulfil their respective roles. In this model, each major functional component operates as an intelligent agent capable of perceiving its environment, making decisions, and communicating with peer agents to achieve coordinated media delivery outcomes.

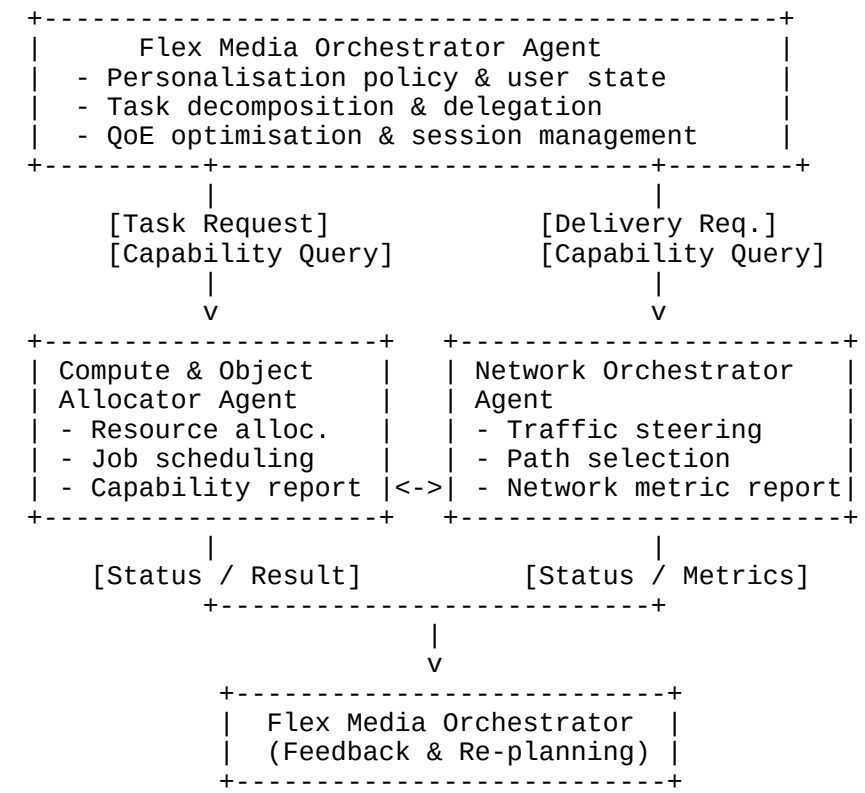
Three primary agent entities map directly onto the existing OBM functional architecture:

* Flex Media Orchestrator Agent: Acts as the top-level orchestration agent. It accepts user requests, applies personalisation policies, decomposes delivery goals into subtasks, and delegates work to subordinate agents. It maintains awareness of user preferences, session state, and editorial constraints, and is responsible for overall QoE optimisation.

* Compute and Object Allocator Agent: Operates as a resource management agent. It receives task assignments from the Flex Media Orchestrator Agent, negotiates compute resource allocation across distributed nodes, tracks job status, and reports resource availability back to the orchestrator. It may spawn, reassign, or terminate compute jobs in response to changing conditions.

* Network Orchestrator Agent: Functions as a network-aware delivery agent. It receives content delivery requirements from the Orchestrator and Allocator agents, reports on current network conditions (latency, bandwidth, congestion), and steers traffic to optimise media object delivery to end users, in coordination with the CATS framework.

The following diagram illustrates the agent entities and their communication relationships:



Agent-to-agent (A2A) communication between these components requires a structured protocol supporting task delegation, capability advertisement, status reporting, and negotiation. Key requirements for such a protocol in the OBM context include:

* Capability Discovery: Agents must be able to advertise and query the capabilities of peer agents, such as available compute types, supported media object formats, or current network path characteristics, so that the orchestrator can make informed delegation decisions.

* Task Delegation and Confirmation: A structured request-response mechanism is needed for the Flex Media Orchestrator Agent to delegate processing or delivery tasks to subordinate agents, and to receive acknowledgement, rejection, or a counter-proposal based on current resource availability.

* Asynchronous Status Reporting: Agents must be able to asynchronously report on the progress and outcome of delegated tasks, including compute job completion, quality degradation events, or network condition changes that adversely affect QoE.

* Negotiation and Conflict Resolution: When multiple agents contend for shared resources, or when QoE constraints cannot be simultaneously satisfied, a negotiation mechanism is needed to re-prioritise tasks and adapt the delivery strategy in real time.

* Security and Authorisation: Agent interactions must be authenticated and authorised to prevent unauthorised task injection or resource manipulation, particularly in multi-operator or federated deployment scenarios.

Emerging AI agent communication frameworks, such as the Model Context Protocol (MCP) and Agent-to-Agent (A2A) protocols, represent candidate mechanisms for structuring these interactions. However, their integration with existing IETF transport, application-layer, and media delivery standards (including CATS-based traffic steering and the media transport protocols described in Section 3.5) requires further investigation.

5. General Metrics

In traditional routing systems, often network path costs may not change frequently unless there is a resource failure or planned outage, whereas network traffic engineering metrics, such as available bandwidth, may fluctuate more dynamically. However, the computation-oriented metrics relevant to OBM can be highly variable, influenced by factors such as session numbers, CPU and GPU utilisation, and memory consumption. Determining the appropriate interval or triggering events for distributing this information is critical, as overly frequent updates may cause unnecessary signalling overhead.

OBM requires the ability to dynamically assess compute availability and adjust media object delivery accordingly. Depending on the decision logic associated with OBM service delivery, one or more compute-related metrics must be conveyed within a CATS domain. The frequency of such conveyance must be optimised to ensure that signalling overhead does not introduce additional network congestion.

While existing routing protocols can provide a baseline for conveying such metrics, alternative mechanisms may be required to efficiently integrate compute-aware decision-making processes.

Furthermore, an effective OBM system should balance network path selection with the real-time availability of compute resources to ensure optimal QoE. This may involve leveraging distributed compute resources across the network, matching computing workloads to resource-availability and allowing OBM elements to be processed closer to the user when necessary. Mechanisms for synchronising compute-aware decisions across different network segments will be crucial to ensuring seamless media composition and delivery.

The categories of metrics relevant to OBM in a compute-aware traffic steering context include:

* Compute and GPU Resource Availability: CPU and GPU utilisation, memory consumption, and storage capacity at different compute nodes.

* Session and User Load: Number of concurrent media sessions, user distribution, and geographic density of active users.

* Processing Latency Delays introduced by encoding, decoding, and media object composition at various compute locations.

* Network Throughput and Congestion: Available bandwidth, packet loss, and jitter affecting media object transmission.

* Edge and Cloud Resource Allocation: Distribution of OBM processing tasks between central cloud servers and edge computing nodes to balance performance and latency.

5.1. Applicable Metrics

In addition to metrics for assessing compute suitability and availability, metrics are also required to select a particular compute site. This decision requires an understanding of both the cost of offloading and its impact on Quality of Experience (QoE). For most flex media experiences, there are also baseline QoE constraints that need to be factored in this decision. These metrics can be grouped in three main classes:

5.1.1. Delivery Performance

This includes metrics associated with video delivery and media experience responsiveness , if the user is afforded agency via interaction:

* Frame Rate: target frame rate for the experience (30 fps for video animations, sampling rate for audio); also specifies the rate of work done required to produce the frames i.e. at frame rate 60 fps, twice the amount of computation is required compared to a 30 fps offloaded experience.

* Frame Size: size of video frames (pixels) or audio samples (bit depth); usual Flex Media video resolutions are 720p, 1080p (HD) and 2160p (4K).

* Bit Rate: amount of media data processed per second. Usually calculated as Frame Rate x Resolution (bps)

* Delay: The end-to-end delay for streaming offloaded flex media consists of the following:

  - Render Delay: time to render each frame; this is dependent on the frame size, the number of objects to render and metrics indicating compute type (CPU, SIMD, GPU) and capabilities (frequency, boosted frequency, number of cores, number of render units, memory bandwidth, memory size, memory utilization, core utilization).

  - Encode Delay: time taken to encode and package frames for streaming. This depends on encoder type (hardware or software), encoding type (some are more optimised for low-latency) and media segment length.

  - Transport Delay: the network propagation delay for a media frame; depends on Frame Size and network bandwidth

5.1.2. Client QoE

These metrics concern the consumption of flex media and the perception of degradations by the user. Some metrics like delay/asynchrony tolerance are set by editorial guidelines. For example, there are different acceptable delays for interactive TV applications like switching media objects than for game-like or XR applications.

* Delay tolerance: The threshold for the particular experience beyond which the delay becomes perceptible/irritating to the user; usually editorially set as it may be experience-specific.

* Object Asynchrony: the asynchrony between the various objects being assembled; offloading one or more objects processing may result in the objects arriving out of sync at the point of assembly due to network delay variation.

* Asynchrony Tolerance: the threshold beyond which the time shift between the objects becomes perceptible to the user. Lip-sync (between audio and video objects) has a lower asynchrony tolerance than Picture-in-Picture.

* Object Quality: the average Frame Rate and the Frame Size of the media object; objects can be streamed at different qualities to meet bandwidth constraints based on user perception.

* Quality Switches (Magnitude, Frequency): if the media object is delivered via adaptive streaming, then this metric measures the frequency and magnitude of switches between available object qualities (e.g. resolution/bit rates).

* Number of Rebuffering Events: number of audio/video stalls over a time period due to network congestion or server performance delaying arrival of frames for decoding and playback.

* Playback Rate: the rates at which different objects are played at the point of assembly and presentation. These can vary if local playback adaptation algorithms are used to overcome object asynchrony.

5.1.3. Cost

These metrics refer to the cost of running offloaded media processing jobs at a selected compute site and streaming the results back to the client.

* Render Cost: this is determined from resources used (CPU/+GPU), the time taken to render, and time to encode a given task for transport.

* Cache Recency: this specifies the (caching policy) i.e. the priority to be set for keeping a generated object frame in a cache. Utilisation of a cache reduces compute resource utilisation. An object with higher Cache Recency indicates a higher probability that this object will be required by another client in the lifetime of the experience.

5.2. Compute Metrics

These metrics are used to assess the suitability of compute resources and their availability for offloading of flex media compute tasks. They denote different types of compute hardware as well as their level of utilisation. GPUs will run tasks such as rendering complex images, where NPUs are preferred for repetitive and less complex AI tasks, such as background blurring or object detection.

* Compute Type: Type of processor (CPU, GPU, Frequency, FLOPS, integer, FP8, 4 octets)

* CPU: Frequency, number of cores, core utilization, memory bandwidth, memory size, memory utilization, power consumption.

* GPU: Frequency, number of render units, memory bandwidth, memory size, memory utilization, core utilization, power consumption. For

* NPU: TOPS , utilization, power consumption

* System Load Average: A measure of the average workload of a system over a time period, providing a snapshot of overall system performance.

* Storage: Available space, read speed, write speed.

5.3. Network Metrics

These metrics enable the assessment the suitability ot network links based on their characteristics as they can adversely affect QoE.

* Latency: The time it takes for data to travel from source to destination is critical for the time-sensitive delivery of flex media.

* Bandwidth: The maximum rate of data transfer across a network path, indicating the capacity of the network link.

* Packet Loss: The percentage of packets that fail to reach their destination, affecting the network connection quality.

* Jitter: The variability in packet delay can impact the performance of real-time applications like VoIP or video streaming.

* Throughput: The actual data transfer rate achieved can be lower than the available bandwidth due to various factors like congestion.

* Error Rates: The rate of erroneous packets, indicating the quality of the network link.

6. Flex Media Service Types

OBM enables the decomposition of traditional (linear) media content into discrete media objects that can be dynamically selected, assembled, and delivered based on user preferences, device capabilities, and network conditions.

This approach enhances personalization, adaptability, and efficiency in media delivery. THere are specific key OBM service types, including:

* Narrative Versioning: enables the dynamic adaptation of media content by allowing multiple versions of a narrative to be encoded as separate media objects. These objects can be selected and assembled in real-time based on user preferences, accessibility requirements, or contextual factors such as location, time of day, or device capabilities.

* Layered Compositing: involves dynamically assembling media content by overlaying multiple independent media layers. This approach allows for real-time customization and modification of media streams without requiring pre-rendered compositions.

* Rendered Objects: refers to media elements that require computational processing before playback. These objects are dynamically generated based on real-time conditions, user interactions, or AI-based content synthesis.

* Non-Graphical Objects encompass metadata, control logic, interactivity triggers, context-aware overlays, and sensory elements (e.g., haptic feedback, spatial audio cues). These objects facilitate context-aware and interactive media experiences.

These four service types form the foundation of OBM and enabling personalized, context-aware, and interactive media experiences. These capabilities align with emerging trends in in-network computing, edge processing, and AI-driven content adaptation for next-generation media applications.

7. Compute and Bandwidth Estimates for Flex Media

A single compute node can handle approximately 60 users for 4K content at 30Hz and 1,200 users for HD content at 30Hz.

Compute Estimates: Narrative Versioning

Number of Users	Worst Case	Best Case (90% Cache Hits)
10,000 users	45 nodes	5 nodes
250,000 users	1,125 nodes	113 nodes
1,000,000 users	4,500 nodes	450 nodes

Compute Estimates: Layered Compositing

Number of Users	Worst Case	Best Case (90% Cache Hits)
10,000 users	111 nodes	84 nodes
250,000 users	2,775 nodes	2,082 nodes
1,000,000 users	11,100 nodes	8,325 nodes

Compute Estimates: Rendered Objects

Number of Users	Worst Case	Best Case (90% Cache Hits)
10,000 users	138 nodes	104 nodes
250,000 users	3,450 nodes	2,600 nodes
1,000,000 users	13,800 nodes	10,400 nodes

Compute Estimates: Non-Graphical

Number of Users	Worst Case
10,000 users	10 nodes
250,000 users	250 nodes
1,000,000 users	1,000 nodes

Bandwidth requirements vary based on content resolution, the number of simultaneous users, and the type of media being delivered.

Bandwidth for Different Resolutions:

* HD (1080p) at 30Hz: A typical bitrate for HD streaming at 30 frames per second is around 5-8 Mbps.

* 4K at 30Hz: Streaming 4K content at 30 frames per second usually requires a bit rate of 15-25 Mbps.

HD Bandwidth Requirements per # of Users

Type	Number of Users	HD Bandwidth Req. (Mbps)
Narrative Versioning	10,000	25,000 - 40,000
	250,000	625,000 - 1,000,000
	1,000,000	2,500,000 - 4,000,000
Layered Compositing	10,000	25,000 - 40,000
	250,000	625,000 - 1,000,000
	1,000,000	2,500,000 - 4,000,000
Rendered Objects	10,000	25,000 - 40,000
	250,000	625,000 - 1,000,000
	1,000,000	2,500,000 - 4,000,000

4K Bandwidth Requirements per # of Users

Type	Number of Users	4K BW Requirement (Mbps)
Narrative Versioning	10,000	75,000 - 125,000
	250,000	625,000 - 1,000,000
	1,000,000	2,500,000 - 4,000,000
Layered Compositing	10,000	25,000 - 40,000
	250,000	625,000 - 1,000,000
	1,000,000	2,500,000 - 4,000,000
Rendered Objects	10,000	25,000 - 40,000
	250,000	625,000 - 1,000,000
	1,000,000	2,500,000 - 4,000,000

Total (HD and 4K) Bandwidth Requirements per # of Users

Type	Number of Users	Total Bandwidth (Gbps)
Narrative Versioning	10,000	100 - 165
	250,000	2,500 - 4,125
	1,000,000	10,000 - 16,500
Layered Compositing	10,000	100 - 165
	250,000	2,500 - 4,125
	1,000,000	10,000 - 16,500
Rendered Objects	10,000	100 - 165
	250,000	2,500 - 4,125
	1,000,000	10,000 - 16,500

8. Scalability Considerations

Scalability concerns for OBM distribution at scale are significant due to the inherently complex and dynamic nature of delivering personalised, interactive content to a wide audience. These concerns primarily revolve around the following aspects:

* Network Bandwidth: Object-based media, by its nature, can require more data transmission than traditional linear media because it involves sending multiple media objects (and potentially multiple versions of each object) to allow for user customisation and interactivity. This can lead to increased bandwidth demands, particularly during peak usage times, posing challenges for content delivery networks (CDNs) and end-user internet connections.

* Compute Resource Usage: To be discussed in later versions of this document.

Addressing these scalability concerns requires innovative solutions in content distribution architectures, such as more intelligent edge computing frameworks, advanced caching strategies, more efficient encoding techniques, and the development of new standards and protocols designed to support the dynamic nature of object-based media. Additionally, leveraging advancements in network infrastructure, such as 5G and beyond, can provide the high bandwidth and low latency needed to deliver personalised, interactive media experiences to large audiences.

8.1. Server-Side Processing

The dynamic assembly of media objects based on user preferences, context, or device capabilities can impose significant processing loads on servers (compute nodes), especially for live or real-time content. Scalability challenges arise in efficiently managing these computational demands, particularly when serving a large and concurrent user base.

8.2. Client-Side Processing

The variability in client devices (ranging from smart TVs and set-top boxes to smartphones and tablets) poses challenges in ensuring a consistent and seamless user experience. Scalability issues may arise from the need to adapt real-time content to each device’s capabilities, considering factors such as processing power, screen size, and available bandwidth.

8.3. Quality of Service (QoS) and Quality of Experience (QoE)

Maintaining high QoS and QoE levels as the user base scales is critical. This includes challenges related to minimising buffering, ensuring synchronisation between media objects (e.g., audio tracks with video), and adapting to varying network conditions in real time.

9. Security Considerations

Ensuring security, privacy and user confidentiality in flex media requires careful management of compute-related and object (asset) information. Exposing details about compute and asset resources to the network may inadvertently reveal sensitive application or domain-level data. To mitigate this risk, strategies for protecting sensitive information should be incorporated into deployments.

Further discussion on this topic will be required.

10. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <https://www.rfc-editor.org/rfc/rfc3550>.

[RFC7826] Schulzrinne, H., Rao, A., Lanphier, R., Westerlund, M., and M. Stiemerling, Ed., "Real-Time Streaming Protocol Version 2.0", RFC 7826, DOI 10.17487/RFC7826, December 2016, <https://www.rfc-editor.org/rfc/rfc7826>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8216] Pantos, R., Ed. and W. May, "HTTP Live Streaming", RFC 8216, DOI 10.17487/RFC8216, August 2017, <https://www.rfc-editor.org/rfc/rfc8216>.

Appendix A. IANA Considerations

This document has no IANA actions.

Acknowledgments

This work has benefited from discussions within the IETF CATS working group community, especially with Adrian Farrel and Peng Liu.

Additionally the work has been partly funded by the UK AI4ME project.

Authors' Addresses

Rajiv Ramdhany
BBC
Email: rajiv.ramdhany@bbc.co.uk

Nicholas Race
Lancaster University
Email: n.race@lancaster.ac.uk

Daniel King
Lancaster University
Email: d.king@lancaster.ac.uk