# APPROACHES TO INTEGRATING CDN TECHNOLOGIES INTO CLASSICAL CABLE VOD PLATFORMS

Charles Hasek
Principal Architect
Time Warner Cable

Santosh Krishnan
Vice President, Product Strategy
Verivue

*Abstract*

*Classical cable video on-demand (VOD) systems have been based on traditional client-server architectures, in which content is replicated on several streaming servers in each geographical location. More recently, cable operators have turned their attention to distributed content delivery networks (CDN) as a solution for expanded content libraries that can no longer be economically addressed by replicated client-server systems. In this paper, we provide justifications for embracing an open standard-based CDN architecture, based on a foundation of HTTP and caching, i.e., technologies that are now widely recognized to have scaled content delivery over IP-based networks. A challenge that remains for classical VOD delivery is to adopt the benefits of such CDN technologies without fork-lift upgrades of the entire existing ecosystem. Here we enumerate a few key modifications to existing components that can enable the creation of hierarchical cache-based architectures. In summary, the proposed modifications can be used as a practical recipe for integrating existing VOD ecosystems into an HTTP-based CDN.*

## INTRODUCTION

Classical cable video on-demand (VOD) systems were constructed using a traditional client-server approach. In essence, this paradigm consisted of silos of content storage and streaming within each geographical area, typically defined as a head-end. The VOD ecosystem, broadly consisting of streaming servers, storage, back-office software and application portals, would then be replicated from head-end to head-end, essentially replicating content libraries at each location. As content offerings have grown, so has the amount of storage at the edge of the networks.

Because of such a silo-based approach, content is propagated (or "pushed") and replicated at each location, regardless of the number of views or popularity. Such replication consumes an inordinate amount of storage, power and space to store very low-use such as unpopular content and long-tail catalogs. Hierarchical storage architectures would allow for much more content to be stored in very inexpensive storage at centralized locations and only moved to servers at the edge when needed, i.e., based "on demand."

A couple of key technology drivers in the content delivery network (CDN) space are now being utilized in the classical VOD space to enable efficient growth of storage and streaming capacities (e.g., see [1], [2]). The first key component is the usage of standard HTTP for content requests and propagation [3]. In fact, in recent years, HTTP-based protocols are already being used to deliver video content end-to-end, in both real-time streaming (e.g., HTTP live streaming [4]) and progressive download fashions. Such capabilities can be applied towards classical VOD delivery as well. A second key component is use of hierarchical caching components and models to efficiently store and move content. This capability can, once and for all, end the storage proliferation issue associated with traditional architectures.

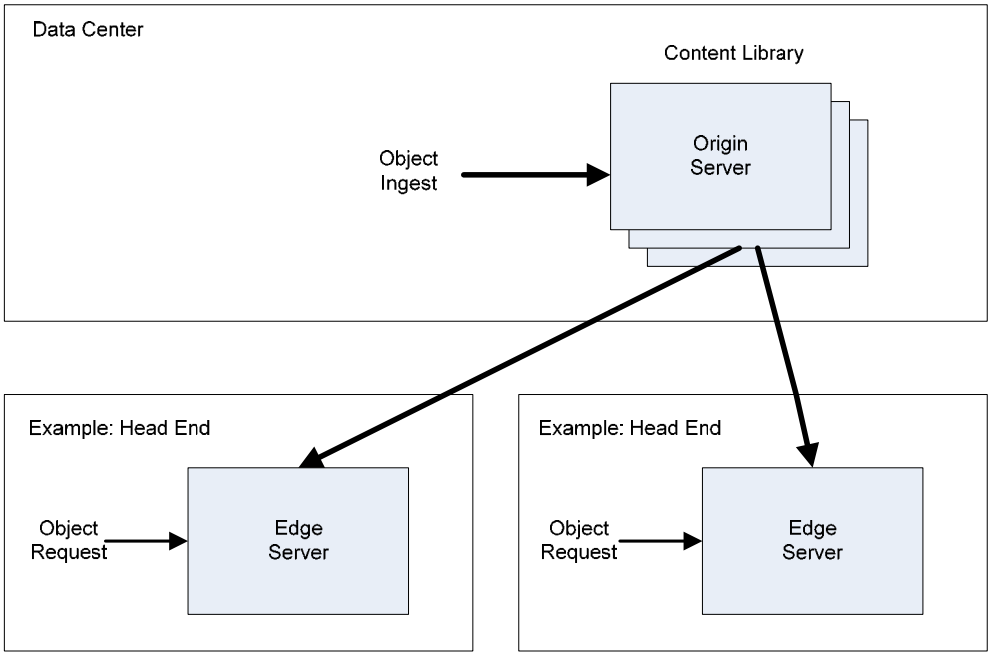


**Figure 1: Example of a two-tier hierarchical content delivery network**

The adoption of CDN technologies in classical VOD itself is not new. However, as cable companies look to deploy this technology, one has to consider the trade-offs between a complete re-architecture of the network to build a best-of-breed HTTP CDN from scratch and building less-than-standard VOD-specific distribution networks with no leverage beyond classical VOD. In order to resolve such trade-offs, it is important to look at existing components that may be modified to be integrated into hierarchical cache-based architectures. Components such as the back-office and streaming servers will need minimal but critical modifications to support such an architecture.

The purpose of this paper is to discuss use-cases and architectures where the above technologies, namely HTTP and caching, can be utilized to create a highly scalable VOD platform. As part of the discussion, the paper will explore advantages that can be gained with very little modification to existing platforms. The proposed modifications may then be used as a practical recipe for integrating existing VOD platforms into a HTTP CDN.

In the rest of the paper, we first discuss the underlying technology drivers, namely HTTP and caching, and the VOD ecosystem components that need modifications to benefit from those drivers. The, we will discuss details and use-cases of how such enhanced VOD ecosystems interface with an HTTP-based CDN, as well as the enhancements themselves.

In the following section, we describe the VOD-specific design considerations, i.e., how to bring the well-known classical VOD application into the CDN model.

## TECHNOLOGY DRIVERS: HTTP AND CACHING

A primary goal of a content delivery network is to scale content libraries by utilizing distributed caching as opposed to replicating entire libraries at each serving location. Figure 1 illustrates a two-tier

hierarchical CDN for reference. The intent of such a CDN is to enable caching of the most popular objects in the edge servers, using optimum caching techniques, and to enable the transfer of content between the origin server and edge server, as popularity changes, utilizing well-chosen transfer techniques.

We posit that content delivery networks represent a strategic infrastructure investment for operators, a layer 7 interconnect for transfer of objects akin to a layer 3 interconnect for transfer of IP packets. Therefore, a careful choice needs to be made on transfer and caching techniques, a choice that lays the foundation for multiple content delivery applications. From that viewpoint, bringing CDN technologies to address classical cable VOD applications is more about bringing VOD ecosystems to a well-chosen CDN infrastructure than the other way around. In other words, it makes economical sense to avoid designing content delivery networks specifically for VOD.

Content Transfer

We propose a standard HTTP-based [5] content delivery network as the foundation for integrating VOD ecosystems, in accordance with principles of Representational State Transfer (REST) [6]. In other words, we narrow down on a standard usage of HTTP for our choice-of-content transfer technique, a choice that has been proven to scale the internet across various content delivery applications. Existing VOD ecosystems form the periphery of a core HTTP CDN. For example, in the figure above, edge servers are VOD streamers, repurposed as caches that employ HTTP to fetch content from multiple tiers of HTTP caches (only one tier, the origin server, is shown in the figure). Object ingest and object request commands are repurposed VOD back-office commands. The following summarizes some aspects of the choice-of-standard HTTP for content transfer:

- *Naming:* VOD objects (media files, metadata) are named using universal resource identifiers (URI)

- *Client intelligence:* The client of the CDN, i.e., the VOD streamer, retains all intelligence regarding when (e.g., cache miss, background fill) and how (e.g., entire files, blocks) to make HTTP requests

- *Media awareness*: All media awareness is encompassed at the peripheries of the CDN. For example, random-access, trick-modes and other rich-media operations are facilitated by the client and the origin. Manifest or index files may be used by clients to make HTTP requests in such a fashion that the core of the CDN remains media unaware.

- *Limited use of extended headers*: Extended HTTP headers should not be used as object modifiers, but may be used in a limited fashion to facilitate auxiliary tasks, such as authentication and bandwidth allocation.

- *Standard DNS/HTTP-based request routing:* Request routing, i.e., determining which specific node in the CDN responds to a VOD streamer request for objects, is a natural consequence of resolving a virtual host name.

REST, in essence, denotes an architectural style that imposes a set of constraints (on the usage of HTTP in this case) to induce desired architectural properties. The desired properties here include keeping the core CDN unencumbered by VOD media specifics to allow unrestricted scale and extensibility, and to maintain cacheability of generic named objects without the need for special application logic. Some of the key REST

constraints most applicable to integrating VOD ecosystems include:

- *State:* VOD streamers, i.e., clients of the CDN, maintain session state, while server nodes within the CDN remain stateless. To enable such statelessness, all requests into the CDN are self-describing (using merely the URI of the desired object) and idempotent, i.e., the order of requests does not modify the identity of the returned objects.

- *Layering:* Components interact only with their immediate neighbors, thus virtualizing the rest of the network. For example, VOD streamers only interact with their immediate parents in the CDN. This kind of layering enables all kinds of clustering, load balancing, redirection, and hierarchies to be hidden from the client.

Caching Techniques

HTTP-based content delivery networks primarily employ distributed caching nodes that pull content on a cache miss, directly from origin servers or from intermediate caches en route to the origin server (multi-tiered cache architectures). This allows caching intelligence to be retained in the clients (the requesting entity) as opposed to centralized tracking systems that may have difficulty scaling as the same CDN is employed for multiple applications. The centralized origin servers provide a "single point of ingest" to push or place content into the CDN. The rest of the CDN then leverages pull-based distributed caching, with algorithms in each node determining the subset of the content library that finds itself in the cache at each given point in time.

A pull-based approach does not restrict the specific caching algorithm that a node may employ to optimize its cache-hit ratio. Some broad examples include:

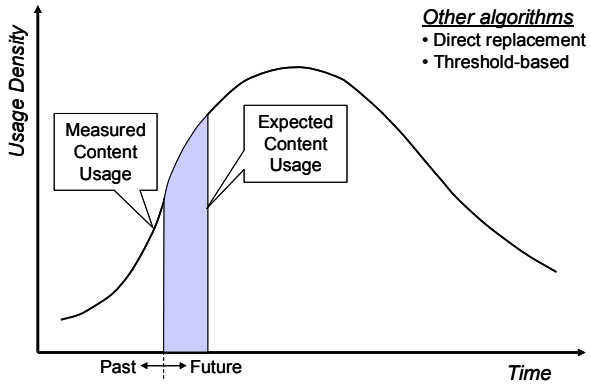


**Figure 2: Trend analysis based caching**

- *Direct replacement:* Every cache-miss leads to filling the cache with the requested object and the eviction of a well-chosen object already in the cache. Examples of eviction policies include the well-known least-recently-used (LRU), least-frequently-used (LFU), and adaptive replacement caching (ARC), which combines aspects of both. Since every request goes through the cache, this is purely a "stream-through" scheme.

- *Threshold-based replacement:* The *N*-th cache-miss for an object within a fixed interval of time leads to filling the cache with the requested object and the eviction of another. The first $N - 1$ cache misses just result in a proxy ("stream-through") or redirect ("stream-around"), on each occasion, to a parent node.

- *Background replacement using trend analysis*: Based on local access patterns and trend analysis, as illustrated in Figure 2 (a content popularity curve for each object in the library), a cache may request content in the background. In this case, all cache misses result in a proxy or redirect to a parent node.

Irrespective of the caching algorithm used, the actual content transfer (and subsequent cache fill if applicable) may occur in one of several modes. A caching node may request content as continuous portions of files (including entire files), e.g., using byte ranges. Alternatively, it may request content on a segment-by-segment basis, where a segment is defined as any well-defined range of the file. In order to let the client exercise its intelligence so as to optimize its local goals, the CDN itself must not restrict either the caching algorithm or the mode of transfer. The analysis of such algorithms themselves is a rich area of research and is outside the scope of this paper.

We propose that while the content transfer methodology must adhere to industry-wide open standards, caching techniques, including the replacement algorithm and mode of transfer, must be left to specific node implementations so as to promote vendor innovation.

VOD Ecosystems and HTTP Caching

In order to utilize a best-of-breed CDN, in accordance with the guiding principles described above, legacy VOD ecosystems will typically require these key enhancements:

- *Back-office modifications:* Many existing back-office systems have typically presumed a replicated model of deployment, i.e., one back-office instance controls one or more replicated sites that contain the entire content library. Due to this assumption, the back-office carefully orchestrates the ingestion of content to specific servers and the subsequent request of content to the respective servers. This link between content ingestion and delivery, which made sense in classical client-server ecosystems, must be decoupled. For example, instead of explicitly ingesting content on each VOD streamer, the back-office may now provide the URI for the ingested asset. A second set of enhancements may be related to the centralization of the back-office functions, which can allow a complete virtualization of the CDN from the point of view of the back-office.

- *VOD streamers as caches:* Legacy VOD streamers have typically been based on the same presumption as above, i.e., content is explicitly pushed into such streamers prior to delivery. VOD streamers which now form the edge ecosystem to the core HTTP-based CDN must be enhanced to include the content transfer and caching techniques described above.

- *Media-related operations:* Operations such as random-access (based on time or chapter numbers) and trick-mode s must now be supported in the context of a CDN. Typically, this is accomplished by generating the necessary meta-data, e.g., manifest files or index files, which can be used by the entire base of VOD streamers. This in essence may require additional elements to generate such meta-data, and enhancements to VOD streamers in order to use it.

Now, we turn to a discussion of how such enhanced VOD ecosystems interface with an HTTP-based CDN and the enhancements themselves.

CDN Standards

As mentioned above, we strongly espouse the usage of industry-wide open standards for content transfer. Standardization in this area will help with interoperability and an ability to leverage already deployed systems. While

many of the foundational items of such a CDN, namely protocols such as HTTP [5] and DNS [7], have long been standardized, the industry has been lacking in the wider adoption of VOD CDN standards.

A promising new standard specification by the IPTV Interoperability Forum (IIF) may address that gap. The IIF Content on Demand specification [8] defines several reference points between components of an HTTP-based video on-demand CDN. For our purposes, the C2 reference point in the specification provides a template for the content transfer interface between edge streamers and an HTTP CDN. Also relevant is the C2 index file specification that aids streamers to perform media-related operations, such as trick mode. While specification of interfaces is out of the scope of this paper, we note here that C2 is an HTTP interface, including URI and header conventions, to pull content from a network. In line with our goals, the specification allows streamers to exercise any caching algorithm or mode of their choosing.

## VOD CDN: DESIGN CONSIDERATIONS

The goal of integrating CDN technologies into classical VOD platforms is to leverage the capabilities of a best-of-breed HTTP-based CDN architecture for content library expansion, while at the same time, maintaining as much of the current VOD infrastructure as possible. In addition to maintaining infrastructure components, such as streamers and the VOD back-office, media-related functions commonly offered in classical VOD, such as trick mode, must be maintained. Similarly, critical back-office functions such as session management, catalogs and billing support must also be maintained.

Figure 3 illustrates how existing VOD platforms may be migrated to the new architecture. HereVOD streamers accustomed to explicit asset ingest (at each head-end) are repurposed as caches that use HTTP (e.g., IIF C2 reference point) to pull content on-demand from the CDN. The back-office ingest and delivery commands are essentially decoupled by centralizing asset ingest. As part of asset ingest, a centralized asset preparation server generates the necessary metadata and media files required to support media-related operations, such as trick mode. Both media and metadata are ingested into a centralized origin server. Instead of explicit asset ingest into the streamer, the streamer is merely notified of the URI of the ingested asset to be used on a cache miss.

### Streamers as Caches

In traditional VOD deployment scenarios, storage and streaming have been tightly integrated at each head-end, as shown at the top of Figure 3. With the advent of offerings such as network DVR and Start-Over, highly scalable ingest mechanisms were added to these systems to allow for many linear channels to be ingested into the VOD streamers. Each such ecosystem then grows atomically and separately from other VOD ecosystems in the network. As such, with growing storage and ingest capacity requirements, replicated expansion becomes expensive and cumbersome. Consequently, the best use of the current VOD ecosystems is to tie them into a larger CDN network, allowing for expansion of content, as shown at the bottom of Figure 3.

Wherever possible, the best option for an operator is to upgrade (e.g., via software update) the standalone VOD clusters so as to enable them to pull files (or fragments of files) from a hierarchical CDN network using HTTP, e.g., using a protocol such as IIF C2. The existing disk subsystems of the standalone cluster would now function as an efficient caching element at the edge. In some

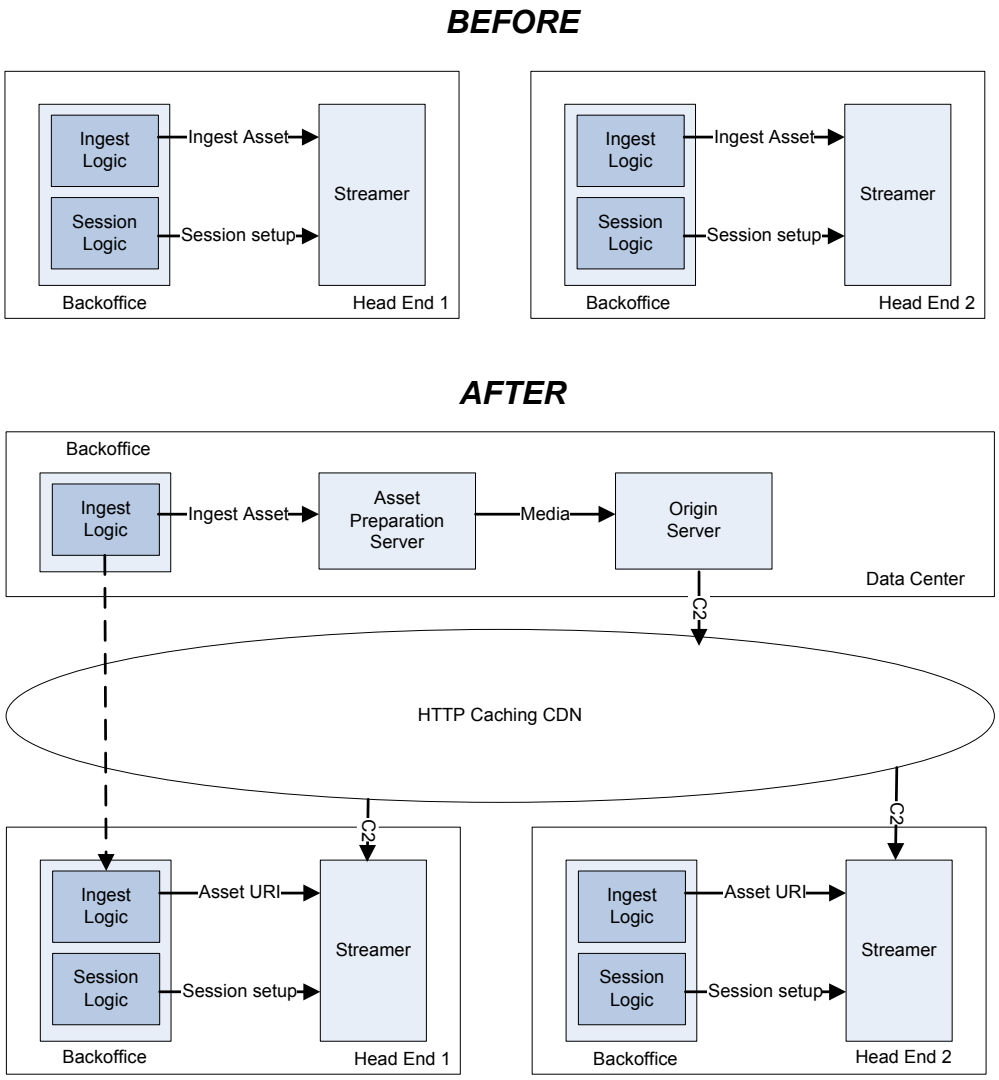


**Figure 3: Classical VOD ecosystem modified for CDN integration**

Cases, depending upon the design of the existing system, storage previously located at the edge may be able to be repurposed towards a central library.

Careful consideration is needed to ensure that the repurposed VOD streamer can handle ingest requirements for an optimized edge cache. For example, if the content library yields an 80-20 popularity curve (80% of requests are of the top 20% of the content library), the streamers must be able to handle an ingest bandwidth of 20% during peak times. Also, the system must be able to support trick mode operations either through real-time processing or via support of common trick mode streams (e.g., as specified by the IIF C2 index file specification). As a cache, the repurposed streamers need to be optimized simultaneously for streaming and ingest performance.

As opposed to proactively ingesting content based upon an ingest command, the VOD streamers must support the ability to provision HTTP-locatable content (URI) to be used to pull content on demand. The VOD streamer would need to be able to make requests to a DNS or HTTP-based request router (e.g., using 302 redirect) to determine

the server location within the CDN responsible for satisfying each on-demand content request.

Well-chosen caching and content life cycle management techniques must be used by the VOD streamers to manage the content (files and file segments) populated as part of the content playback requests. As we have described in the previous section, a number of different caching methods may be used, such as direct replacement with LRU, LFU or ARC eviction policies, to maximize the cache hit ratio. As part of the chosen caching technique, stream-around or stream-through methods may be considered.

The edge VOD complex acts as a termination system for such protocols as Session Setup Protocol (SSP) [9] and Lightweight Stream Control Protocol (LSCP) [9]. By allowing the VOD complex to manage these protocols required for classical VOD, the CDN does not need any awareness of customers, session or state. All these components and states are managed by the edge streamer, thus requiring no changes to the set-top box (STB) client or the rest of the VOD ecosystem. The CDN network is also protected and firewalled from the STB using such an edge VOD system.

Back-office Modifications

A number of back–office systems exist in the cable community today, many of which are built around the popular Interactive Services Architecture (ISA) [9] or Next-Generation On-Demand (NGOD) protocol suites. With a few modifications to such back-office systems, a standalone VOD ecosystem can be integrated into a CDN.

One initial area that needs to be addressed is the ability to provision "CDN" content on the back office and VOD streamers. In the traditional architecture, content is provisioned on streamers via the back office (using an asset ingest command), which immediately loads the content onto the streamers for playback. When using a CDN, the back-office system must instead use indicators (URI) to the VOD streamers to provision the availability and authoritative location of content (e.g., using a modified asset ingest command). The content itself must not be proactively loaded.

One method for achieving this combination of static provisioning and subsequent dynamic retrieval is using an HTTP flag in the asset ingest provisioning process. The presence of the HTTP flag conveys the provisioning of a URI to the VOD streamer for future HTTP-based retrieval from the CDN. In addition to a modified provisioning command to the streamer, the asset management component of the back office is essentially centralized. Assets are provisioned into a central HTTP-based origin server, using either the existing (e.g., ISA ingest) or a new asset ingest command. As part of such provisioning, a central asset preparation server may generate index and trick files, which are also placed into the central origin server (as shown in Figure 3).

The back-office system must also be able to manage larger catalogs, even though all content does not persist at the edge VOD complexes. A related area of note is the required presentation mechanisms for large content catalogs, including a robust navigation system supporting the larger catalog. Techniques such as web-based navigation, play-listing and reservation lists, and tablet/smart-phone navigation apps could all be utilized to allow for additional navigation ease.

The modified back office must also provide the necessary content lifecycle management to ensure that files (or file segments) are properly accounted for and removed as needed (e.g., using CDN-wide

purging of content, if necessary). The back-office must continue to honor updates to license and offering windows.

Beyond catalog management, ingest provisioning and content lifecycle management, the back office does not need to have any awareness of where the actual content files reside. As such, this allows the operator to minimize the amount of changes necessary in the back-office system for CDN integration, thereby simplifying operation and design.

Trick Mode and Media-related Operations

One of the popular features of a classical VOD platform is the ability to fast forward, pause and rewind content, much as with an in-home DVD player. When content is pro-actively ingested into a VOD system, as in the traditional architecture, there are several simple and well-known processes for locally creating trick mode files and/or indexes in support of such features. With CDN integration, the edge streamer may not be able to proactively create trick and index files, since the content is typically not pre-analyzed. This may in some cases lead to a failure to support trick mode features, especially on a cache miss, e.g., a request for a 32x fast forward as content is being streamed through the edge VOD streamer at normal speed. Therefore, it is critical to explore new methodologies to supporting trick modes in a CDN environment.

One method to support trick modes is to provide an index file (constructed by the asset preparation server in Figure 3) that outlines the structure of the content, such as the location of I, P and B frames for MPEG-2 content. Such a file can be small enough to be transferred during the initial phase of the content transfer, thus giving the edge streamer a “hint file” to assemble trick mode streams. The streamer then could pull the appropriate frames, as needed, to build out the trick mode stream on the fly.

Another method relies on pre-generated trick mode files, created centrally (e.g., by the asset preparation server in Figure 3) during the asset ingest process. The edge streamer would then pull the appropriate pre-generated trick mode file, based on the selected trick speed. A companion index file is typically used to correlate files of different speeds. The edge streamer may then cache the trick mode file segments much like the normal-speed files. When using this approach, it is critical to support a standardized trick mode file format for interoperability.

Another potential method that eliminates the need for both index and trick mode files relies on retrieving content faster than real-time and using the existing local process. However, if the cache-miss ratios are expected to be non- trivial, coupled with a non-uniform arrival of trick mode requests during cache misses, this method becomes highly impractical because of a multiplicative effect on the required cache-miss bandwidth, as well as the unnecessary retrieval of portions of files that may never be viewed. Not to mention, the I/O subsystems of many VOD systems may have difficulty maintaining the high ingest rate while attempting to create a trick mode stream to the customer. Multiple transfers at this high ingest rate may cause most disk I/O sub-systems to perform poorly.

From a practical standpoint, an operator should provide as much flexibility as possible since different VOD streamers may employ different options for trick mode support in a CDN. As older servers are aged out and replaced, it could be an opportunity to harmonize methodologies and technologies.

## CONCLUSIONS

The ability to support cloud and CDN technologies for classical VOD delivery platforms is available now and can be leveraged to allow for service growth. We have illustrated how operators can both build upon a foundation of best-of-breed CDN technology and, at the same time, retain existing infrastructure with a few key modifications. It is critical for operators and their partners to jointly develop open and publishable standards to allow for interoperability and for best-of-breed technologies to take hold. A modular approach to design allows the system to grow organically for expanded content offerings, new technology adoptions, and rapid deployment of new products to customers.

## REFERENCES


[1] W. Mao, "Building Large VOD Libraries with Next Generation on Demand Architecture," NCTA Technical Papers, 2008

[2] W. Mao, "Key Architecture and Interface Options for IP Video Over Cable," SCTE Conference on Emerging Technologies, 2009

[3] S. Krishnan, W. Mao, "Open Content Delivery Networks for Managed Video Delivery to Multiple Screens," SCTE Cable-Tec Expo, 2010

[4] R. Pantos, ed., "HTTP Live Streaming," draft-pantos-http-live-streaming-01, Internet Draft (work in progress), Internet Engineering Task Force, Jun 2009

[5] R. Fielding et al., "HyperText Transfer Protocol—HTTP/1.1," Request for Comments (RFC) 2616, Internet Engineering Task Force, 1999

[6] R. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Doctoral Dissertation, Chap. 5, UC Irvine, 2000

[7] P. Mockapetris, "Domain Names – Implementation and Specification," Request for Comments (RFC) 1035, Internet Engineering Task Force, 1989

[8] ATIS IPTV Interoperability Forum, "ATIS-0800042: IPTV Content on Demand Service," Dec 2010

[9] Interactive Services Architecture, www.interactiveservices.org, Time Warner Cable