Scaling IP Advertising Using Manifest Manipulation

A Technical Paper prepared for SCTE•ISBE by

Vipul Patel
Vice President, Advanced Video Engineering
Charter Communications
8560 Upland Drive, Suite B, Englewood, Colorado 80112
+1 833 267 6097
Vipul.Patel@charter.com

Xavier Denis
Director, Product Management, Advertising Solutions
CommScope
1725 NW 167th Place, Beaverton, OR 97006
+1 503 495 9485
Xavier.Denis@CommScope.com

Table of Contents

Title **Page Number**

List of Figures

Introduction

Charter Communications has pioneered the development of streaming technology to augment their wide portfolio of advanced residential broadband services. One of the challenges Charter faced was to maintain a pristine customer entertainment experience while maximizing revenues through targeted advertising.

In order to effectively scale to millions of IP subscribers with billions of ad impressions, Charter undertook a major effort that included:

- Evaluating and selecting between Server-side and Client-side Ad Insertion
- Architecting a solution that can scale horizontally in a distributed environment
- Designing for high availability and resiliency
- Leveraging virtualization and cloud computing models
- Optimizing routing and load distribution
- Adapting to a diverse array of client platforms and Ad Decision Systems (ADSs)
- Enhancing the platform to support robust monitoring and accurate metrics and analytics

This case study provides an insight into the analysis, implementation, and best practices for scaling IP advertising in a demanding Pay TV environment.

Content

1. Market shift

1.1. Transition to IP video

Much has been written about changing consumer habits when it comes to video consumption. The rise of subscription Video on Demand (VOD) services, "cord-cutting", short-form Internet video, Over-the-Top (OTT) linear Pay TV services, and others have all had a dramatic impact on the Pay TV industry.

What consumers may not realize is the underlying technology shift in how video is delivered. In order to leverage the Internet for video delivery and easily support multiple devices, video is increasingly delivered over IP as series of segmented files encoded in multiple bit rates. This process is commonly (and somewhat inaccurately) referred to as Adapative Bitrate (ABR) streaming.

This new technology has not gone unnoticed by traditional Pay TV providers, many of which have long offered IP-based alternatives to their existing delivery models. Originally delivered as companion services such as "TV Everywhere", operators are increasingly offering fully IP-based services targeting the main screen and other devices, both inside and outside of the home.

1.2. Impact on video advertising

This shift is driving changes in the way that video advertising is delivered, sold and evaluated. Just as importantly, it is also creating several technical and operational challenges, some of which will be the focus of this paper.

While advertising in the IP realm requires new approaches from a technology perspective, it offers many potential advantages, including:

- Normally delivered as a unicast service, IP Video enables targeting ads for each individual stream, meaning that ads are more relevant to consumers and advertisers can target any desired audience segment
- IP Video makes each ad view measurable, which helps create new value metrics to monetize more content, including linear channels without traditional ratings
- Leveraging technologies like manifest manipulation, IP Video can interface to modern dynamic ad workflows for more efficient ad campaign execution and less ad fatigue thanks to frequency capping

While the definition of "TV" may be blurring, traditional linear television advertising was still a massive $195B business in 2018 and will grow to $210B by 2023 [1]. The addition of IP technology with its increased targeting capabilities to this powerful platform brings tremendous opportunity. For example, studies have found that ad completion rates on IP-connected TVs were 95%, 32% higher than mobile, and 27% more than desktop [3].

2. Technology changes

2.1. MPEG TS streaming to ABR video

MPEG Transport Stream broadcast delivery over QAM has been the way to reliably deliver video to large audiences for decades. But the advent of streaming technology over IP coupled with intense competition and the large market for streaming devices have disrupted that paradigm. Adaptive Bitrate (ABR) protocols like HTTP Live Streaming (HLS) and Dynamic Adaptive Streaming over HTTP (DASH) have matured to enable robust and scalable delivery while standards efforts have improved the economics of building large streaming systems leveraging operators' IP networks. New standards like Common Media Application Format (CMAF) are aiming to foster more innovation in the industry by promoting common encoding and encryption formats, and by reducing latency in live streaming to parity with broadcast delivery.

2.2. Advertising technology changes (splicer to manifest manipulation)

In the realm of ABR video streaming, manifest manipulation is the rough equivalent of splicing ads into a broadcast Transport Stream delivered over UDP transport. The differences between the two stem from technological advances and the different problem spaces.

Transport Stream delivery needed to cope with rate clamping in order to pass ads without exceeding the constraints of statistically-multiplexed services to "fit" the largest number of services in a 6-MHz channel. ABR delivery deals with matching content and ad representations across a range of bitrate ladders, formats and codecs, and supporting a large array of streaming devices that often behave inconsistently.

The two approaches to the problem of ad insertion in ABR streams are manifest manipulation (sometimes referred to as Server-Side Ad Insertion because the function is deployed in the operator network) and Client-Side Ad Insertion. Manifest manipulation provides a seamless way to replace or insert video chunks in ABR streams by altering the manifests of those streams independent of the client-side device and player technology. Client-Side Ad Insertion involves implementing this logic in the device itself.

While Client-Side Ad Insertion involves the complex logic of chunk and buffer management, integration with Ad Decision Systems, and other backend ad workflows, manifest manipulation is client-agnostic, and avoids duplicating backend integration costs.

These advantages help explain why manifest manipulation is gaining broad industry adoption for ad insertion and a growing number of applications [4].

3. Charter case study

3.1. A review of the IP journey to date and current operation

Charter is one of the largest providers of television, broadband internet, and voice services in the world. Known for its commitment to integrating the highest quality service with uniquely differentiated products, Charter is at the intersection of technology and entertainment, facilitating essential communications that connect more than 28 million residential and business customers in 41 states.

Building a scalable IP video network and service is core to Charter's vision to delight its customers with the greatest choice of content and a superior experience available on any device.

To understand how to approach building a scalable IP video service, it is important to step back and see where it all started. Charter initially marketed its add-on content streaming capability as an augmentation to its set-top box (STB) delivered television product. To that end, the strategy was to focus on the building blocks of the system.

When Charter launched its IP video service in 2015, ABR technology was at various stages of maturation. While HLS was scaling remarkably well on the back of strong market demand for the iPhone®, the MPEG-DASH standard was taking longer to yield robust and mature client implementations that could be deployed at scale. Charter used Microsoft® Smooth Streaming in order to support critical platforms like the Xbox® and Samsung® smart TVs.

From a monetization perspective, Charter chose manifest manipulation because it would:

- Ease support of a constantly expanding set of IP-connected devices such as smart TVs, OTT streaming devices, game consoles, PCs, tablets and phones
- Decouple integration with backend systems like Ad Decision Systems from client integrations
- Provide more control over the quality of the user streaming experience by ensuring clean ad insertions and avoiding rebuffering

The HLS streaming protocol inherently lends itself to manifest manipulation for supporting Dynamic Ad Insertion (DAI) and alternate content switching. For the Smooth Streaming protocol, a simple manifest manipulation is not sufficient as the media segments themselves need to be altered. Alternatively, DAI for Smooth Streaming devices is typically implemented using Client-Side Ad insertion, which poses its own set of challenges related to Ad Decision System integration and maintaining media stream buffers. In the near future, the Smooth Streaming devices can switch to the DASH streaming format, which also supports DAI and alternate content switching through server-side manifest manipulation.

Monetization is only possible with robust reporting. Charter initially experimented with server-side ad beaconing during manifest creation but realized that ad tracking events would often be mis-aligned with the occurrence of the playback events. They determined that building a common cross-client ad beaconing framework allowed them to better conform with industry standards for ad metrics collection.

Over the past two years, Charter has seen session usage grow 4-fold (see below diagram), underscoring the reality that as Charter prepared to evolve its offering to target the primary TV screen in the home, it needed to ensure its platform would continue to scale.

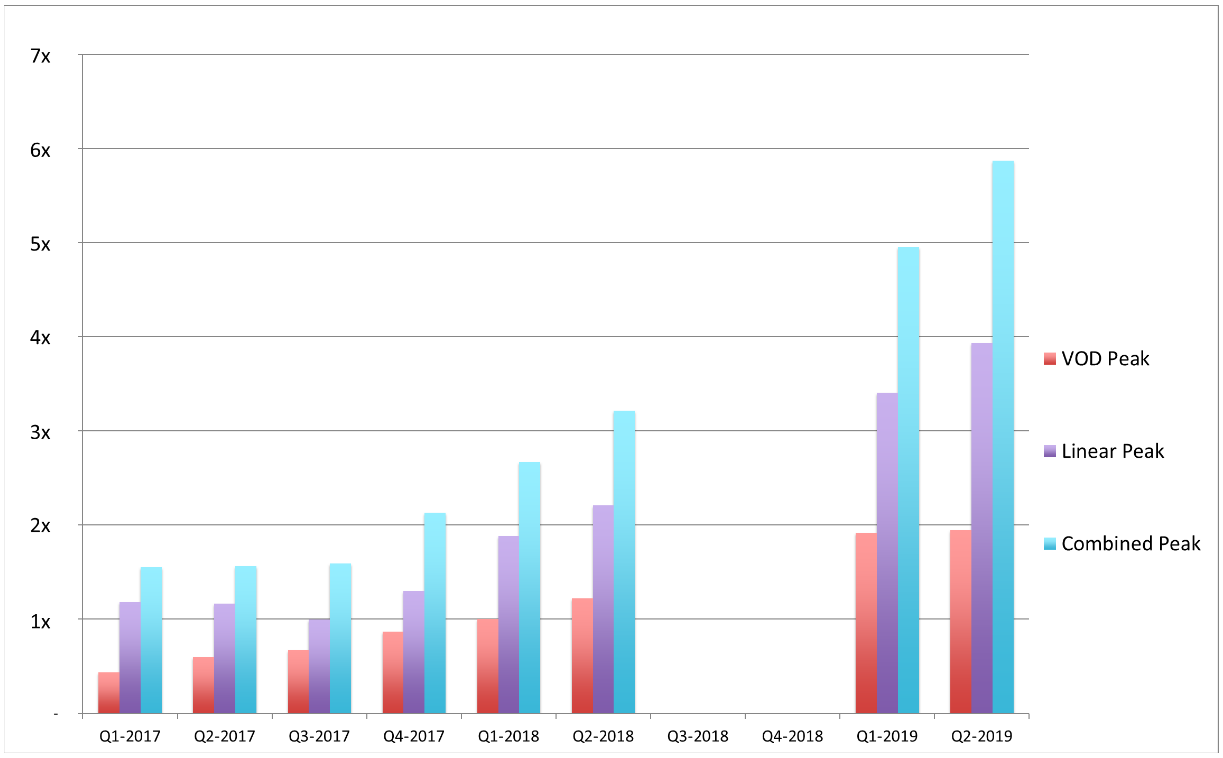

Figure 1 - Charter daily peak concurrent IP video sessions, by quarter

3.2. The 10M-session challenge – Taking IP video monetization to the next step

After the initial successful launch, which extended the primary video services to subscribers on IP-connected devices, Charter focused on updating the platform architecture so that it was scalable and resilient to support IP-based video as the primary service. In doing so, the company was confronted with several challenges.

- Capacity expansion and resiliency
- Performance
- Load distribution
- Moving from an appliance model to a Data Center model
- Player compatibility
- Metrics and monitoring

3.2.1. Capacity expansion and resiliency

As the engine of monetization for IP Video, manifest manipulation was high among Charter's priorities. Charter decided to launch the service with a centralized manifest manipulator design which was a good choice, because it reduced deployment risk, provided for a better ability to contain infrastructure costs, and eased operations.

Since manifests are personalized, they are by nature not cacheable by the Content Delivery Network (CDN), and manifest traffic needs to traverse the network from the manifest manipulator to the clients. As service usage increased, the upward trend in network traffic prompted questions about how to more effectively manage manifest session load at scale. Some have proposed moving manifest manipulation closer to the edge of the network to alleviate this problem [4].

Overall service resiliency, another major factor for Charter, is heavily impacted by where the system is deployed and the type of redundancy it offers. In that regard, a distributed approach provides the benefit of spreading the load across more resources, and isolating the negative impact of any outage. The right solution, however, should ensure the appropriate amount of redundancy and fault-tolerance to minimize revenue impact through fail-over capabilities.

Overall, Charter needed a more flexible way to grow capacity, either centrally or at the edge of the network, with the ability to maintain resilient services.

3.2.2. Performance

Evidence suggests a clear correlation between viewer engagement and video quality issues such as shifting bitrates and video stuttering. Even small playback delays can lead significant portions of a stream's audience to tune out [2]. The ability to monetize IP video and control alternate content events depends in no small part on the manifest manipulator's robustness.

The issues are complex in part because of the different characteristics of live event streaming, scheduled linear television streaming, and VOD streaming experiences. Live event and linear television streaming are highly transactional, with high volumes of simultaneous sessions, and more susceptibility to variations in response times and latency. While VOD is more storage intensive, it imposes less stringent requirements for near real-time responsiveness.

The manifest manipulator is part of an elaborate delivery pipeline of interconnected components (Packager, CDN, and clients). Conditions as varied as packet loss or congestion, clients struggling with slow connections (such as weak Wi-Fi/wireless signals), packager failures, CDN responsiveness, and CDN misconfiguration cause severe quality issues that can be compounded by sensitive clients. Any mechanisms provided by the system to absorb the effect of these conditions will have a positive impact on performance.

3.2.3. Load distribution

Load distribution concerns the imperative of maximizing the use of available resources by evenly distributing sessions. Since we are talking about video, it is clear that we want to also avoid the magnifying impact of requests jumping between different manifest manipulation servers throughout the same session. We will cover the tools and best practices to consider when designing a load distribution model that can scale.

3.2.4. Moving from an appliance model to a Data Center model

As the core video processing functions move to pure software, the Data Center model offers a natural path to scale. While moving away from dedicated hardware toward a common virtualized infrastructure that can be leveraged across applications is a sound approach, advertising for Pay TV video is a specific application with unique requirements. The key challenge is adapting the resource allocation policies to the application to support a constant low latency response to every request on each video stream. Resource sharing among Virtual Machines (VMs) may be possible with certain applications. However, manifest manipulation is a real-time mission critical application that is heavily dependent on compute and network availability.

3.2.5. Player compatibility

Charter is very familiar with the challenge of supporting a vast array of IP streaming devices. The problems range from player device compatibility and stability, to varying player behaviors. Some players start at the lowest profile and ramp up, while others start at a middle bitrate. Some can handle chunks of different durations. Differences also exist in how players calculate buffering requirements in order to determine what bitrate to request for future chunks. These are part of a broad array of issues addressed in [4].

When Charter launched its service on the Apple TV® platform in early 2019, it already had many client integrations under its belt, including Roku®, iOS, and Android devices. Those earlier efforts greatly helped reduce risk and time to deployment for the support of DAI and alternate content switching on this new platform.

This serves to underscore another aspect of delivering IP Video at scale. Given the inevitable proliferation of new devices and client technologies, investing in the testing and integration capabilities to qualify these new platforms plays a key role in their successful launch and support.

3.2.6. Metrics and monitoring

This is a broad topic that covers many aspects of the video service. Addressing these areas impacts revenue directly (failed ad insertions causing ad inventory waste) and indirectly (a bad user experience leading to higher churn rates over time). Some of the relevant questions for Charter included:

- What are the critical metrics to measure overall video quality and ad performance?
- What tools are needed to effectively troubleshoot at scale?
- Where and how does the data get acquired?

Evidence shows that viewer engagement with ads is heavily dependent on the quality of the streaming experience. It is critical to be able to troubleshoot ad failure points (missing ads, invalid manifests, empty ad responses from ADSs, etc.) and correlate those events to their impact on revenue.

4. Anatomy of manifest manipulation

Live ABR streams (HLS and DASH) generated through a transcoder and packager system are made available to clients through a manifest. The manifest updates on a web server and includes a window of media segments available to be played by all downstream clients. Manifest manipulation can personalize these streams by presenting each client the entertainment content while also altering media segments so the client will also play ads, alternate content, emergency alerts, etc.

Manifest manipulation uses features that exist in both the HLS and MPEG-DASH standards to insert or replace the media segments presented to each client. This may be triggered by markers or event tags in upstream manifests indicating the precise splice point or cue for ad breaks (or alternate content). Inserting ads or alternate content is achieved through defined periods or discontinuity markers in manifests that cause the client to prepare the decoding of the new media for seamless rendering across the transitions.

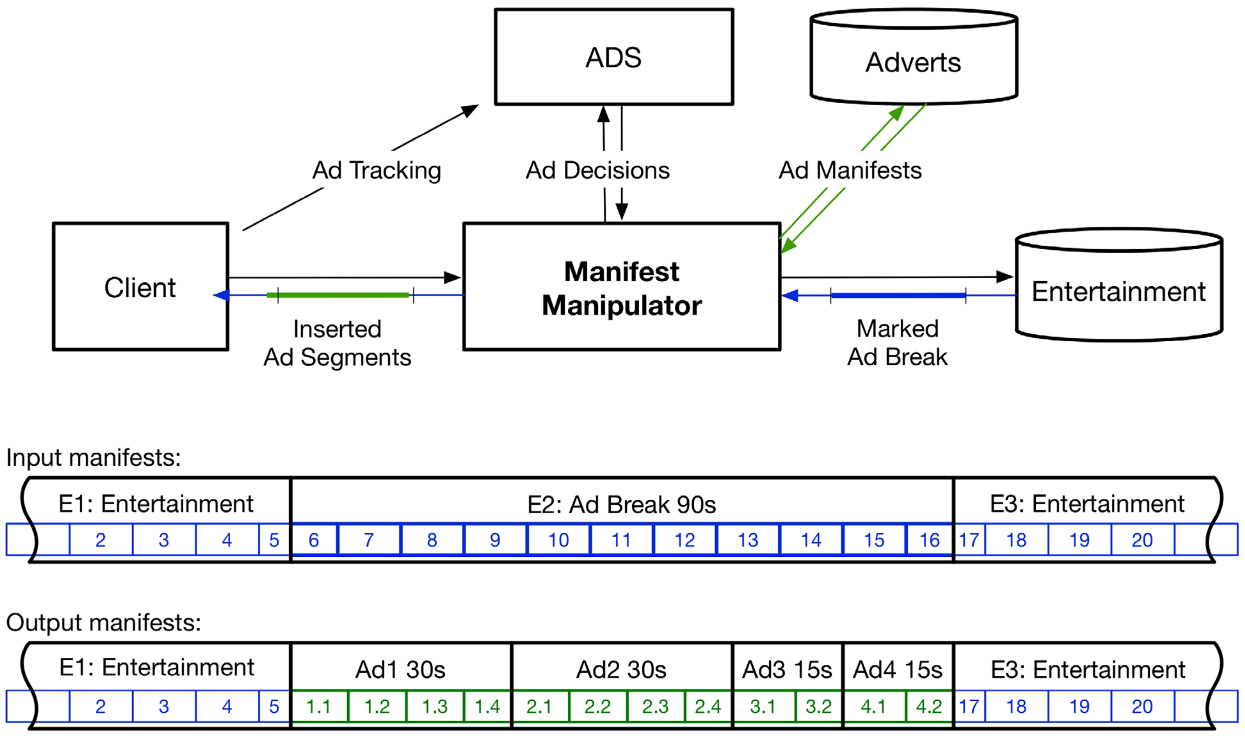

Figure 2 - Manifest manipulation

Each client makes requests to update its manifest asynchronously to when the media segments are first published by a packager. A client typically starts playback rendering a few segments behind the most recent segment to allow tolerance for bandwidth fluctuations and gain the ability to monitor buffer depth in order to adapt media quality to a bitrate that can be sustained by current bandwidth rates.

For manifest manipulation to insert content into a continuous client timeline, it must maintain an appropriate history of past segments in the manifest window while appending new media segments to that timeline. The application must track the rate of the upstream entertainment media in order to output that same rate of inserted media. In some cases, an upstream ad break may signal an early abort, which needs to immediately return to the correct entertainment segment. Handling such a condition involves negotiating the trade-off between staying close to the input manifest's edge and time-accuracy of return to entertainment.

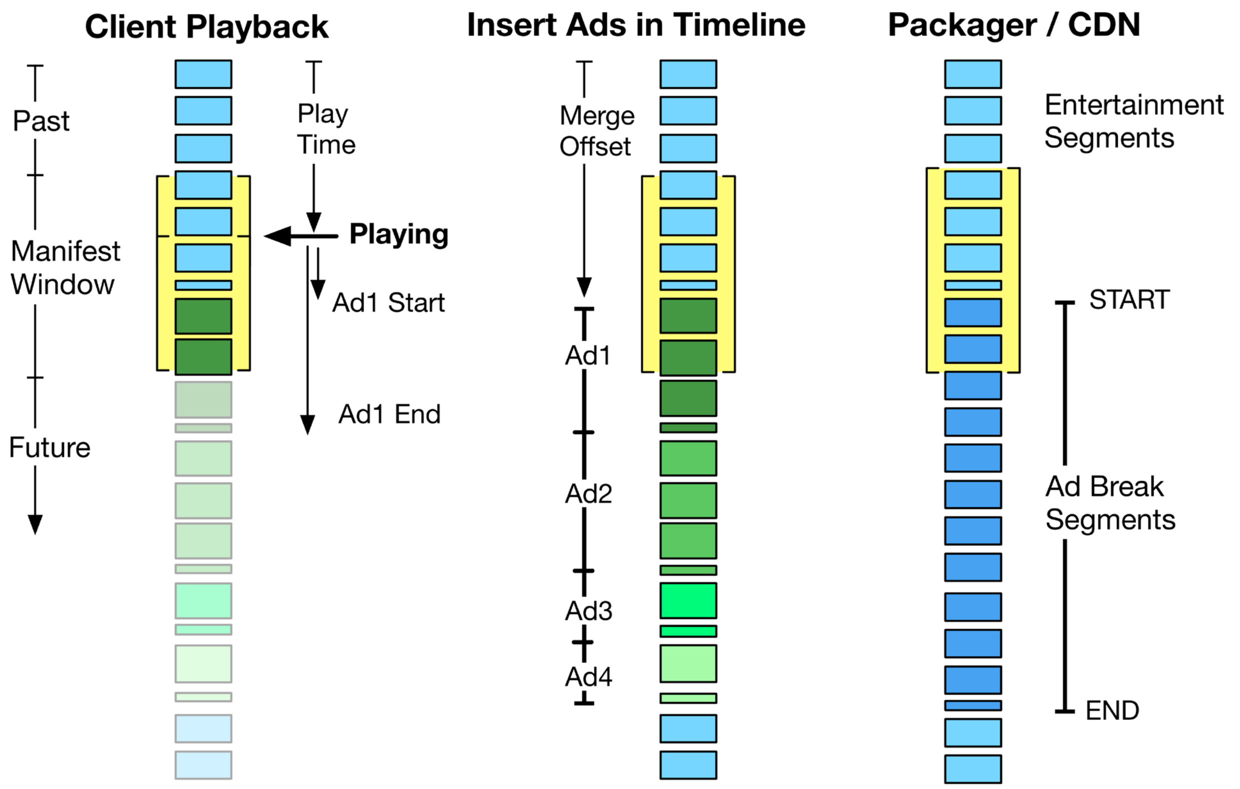

Figure 3 - Manifest lifecycle through delivery pipeline

The manifest manipulator typically executes a workflow defined for a type of session based on attributes of the client and parameters from an initial playback URL. This workflow determines routing policies and settings for Ad Decision Systems (ADS) or Alternate Content Decision Services (ACDS). As each session has signals that determine need for decisions, the initial session attributes can be passed as targeting criteria for these decisions and campaigns. The response from these services determines the content URLs that will be inserted into that session.

A session state object typically caches the session context (ad decisions, alternate content URL, etc.) within the current manifest manipulator instance. That instance can most efficiently build upon the prior output manifests, continue segments from recent placement decision, and maintain tracking of the session variations to deliver seamless playback. The session state also provides opportunity to report on analytics indicating the client targeting attributes, policy workflows, ad opportunities, ad placements, and quality metrics such as shifting quality levels as reported in Figure 2 above.

The manifest manipulation service maintains resiliency of the above session state so, in the rare case the active instance for a session fails or is unreachable, the session can be redistributed and recovered on a neighboring instance.

5. The keys to designing for scale

In response to the challenges described in Section 3.2, Charter identified the following lessons learned in several areas of focus. Section 6 elaborates on each of these topics.

5.1. Horizontal scaling

A. Scale out using an atomic work engine instance that can be tiled outward with no practical limit

B. Preserve flexibility between centralized and distributed deployment archiectures

C. Leverage orchestration tools to ease expansions and contractions to meet dynamic loads

D. Be pragmatic in designing for resiliency and service availability

5.2. Performance and load distribution

A. The Dos and Don'ts of virtualization and infrastructure planning

B. Devise a multi-prong strategy to manage load on the network

5.3. Client Integration

A. Engineering manifest manipulation to adapt to nuanced client behaviors

B. Perform testing, testing, and more testing

5.4. Metrics and Monitoring

A. Define the key telemetry data points and strategy to scale data aggregation and storage

B. Develop the right monitoring tools to find the needle in the haystack

6. How did Charter's investment pay off?

For Charter the key design criteria to build the Next Generation IP video system were:

- Optimize traffic load at the edge of the network to minimize client latency
- Ease the ability to dynamically adapt to fluctuating workloads
- Optimize load distribution across all instances
- Build resiliency to enable lights-out, 24x7 service
- Design for geo-redundancy and 100% service availability
- Enable zero-downtime software upgrades and maintenance

While these criteria applied to several components of the video delivery pipeline, this section focuses on how Charter guided the redesign of the manifest manipulation function and the operational impact that resulted from the changes.

6.1. Leveraging horizontal scaling

The goal was to flexibly accommodate any peak load of VOD and live IP-connected client sessions by easily scaling out the number of work engine nodes in the system without any limitations while preserving full session resiliency and service availability. A second goal was to be able to continually roll out new features via software upgrades without disrupting revenue-generating services supported by the application.

That means we require the ability to effectively manage a large collection of work engines (or "nodes") that operate semi-independently of each other so that it is possible, to:

- Selectively push new software to any subset of nodes
- Turn on or off new nodes on any infrastructure, whether hosted or cloud-based
- Provide session resiliency across instances within a Data Center or across regional ata Centers

The preferred design was, and continues to be, one where session distribution is built in the node, without reliance on a central database (see figure below).

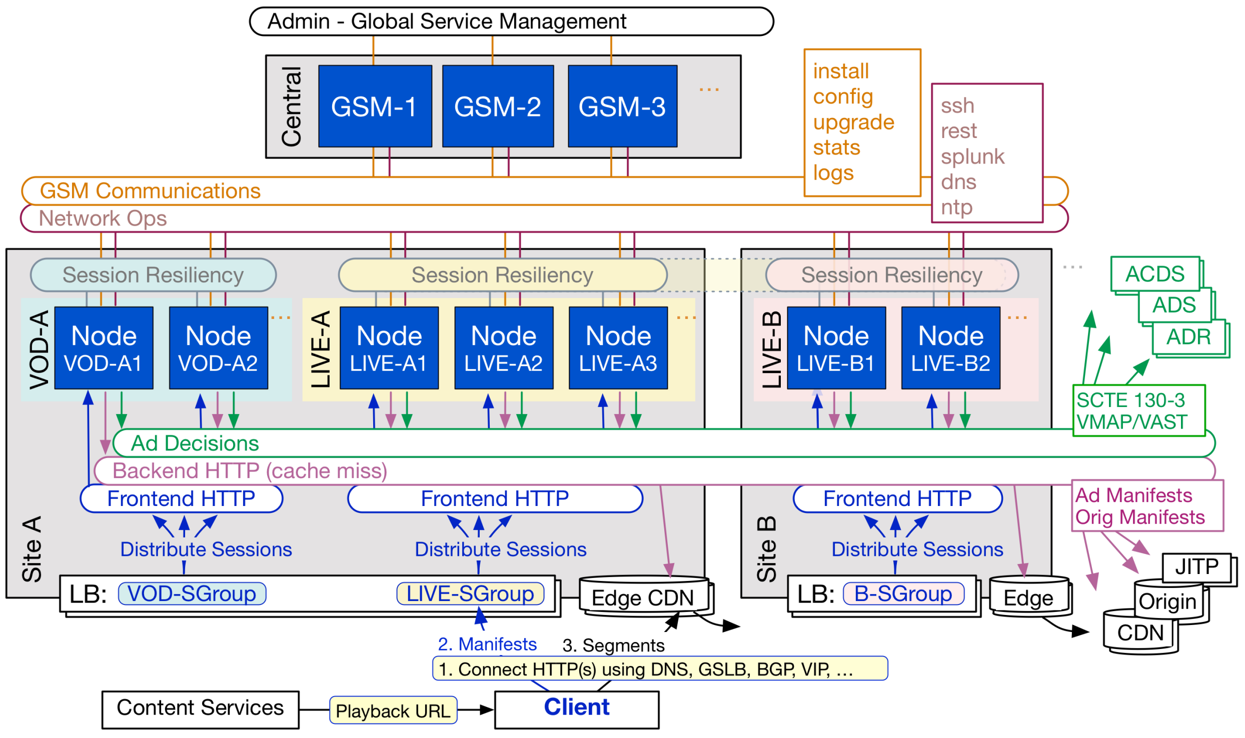

Figure 4 - Horizontally scaling manifest manipulator design

The manifest manipulator operates behind the load balancer (LB), which can support distribution using session affinity such that the node designated at session start keeps getting the requests under normal operations. This ensures maximum efficiency by preserving processing on the node that is in charge of maintaining session state. Session state is also written in a distributed data store that enables "neighboring" nodes to handle requests seamlessly, should the session need to move because of hardware failure, or any cause making the node unresponsive to the load balancer (see Figure 5).

This level of resiliency can be achieved among any number of nodes clustered together as a single set of neighbors within the same site. And it can be extended to support geo-redundancy, wherein two active sets of nodes installed at separate physical locations are sized such that either can handle the peak load as needed, yet load is evenly distributed across both under normal conditions (see Figure 4 illustrating manifest manipulation nodes deployed at sites A and B). The question of which session state resiliency policy is implemented depends on defining the service SLA and evaluating the backend network costs required to meet it.

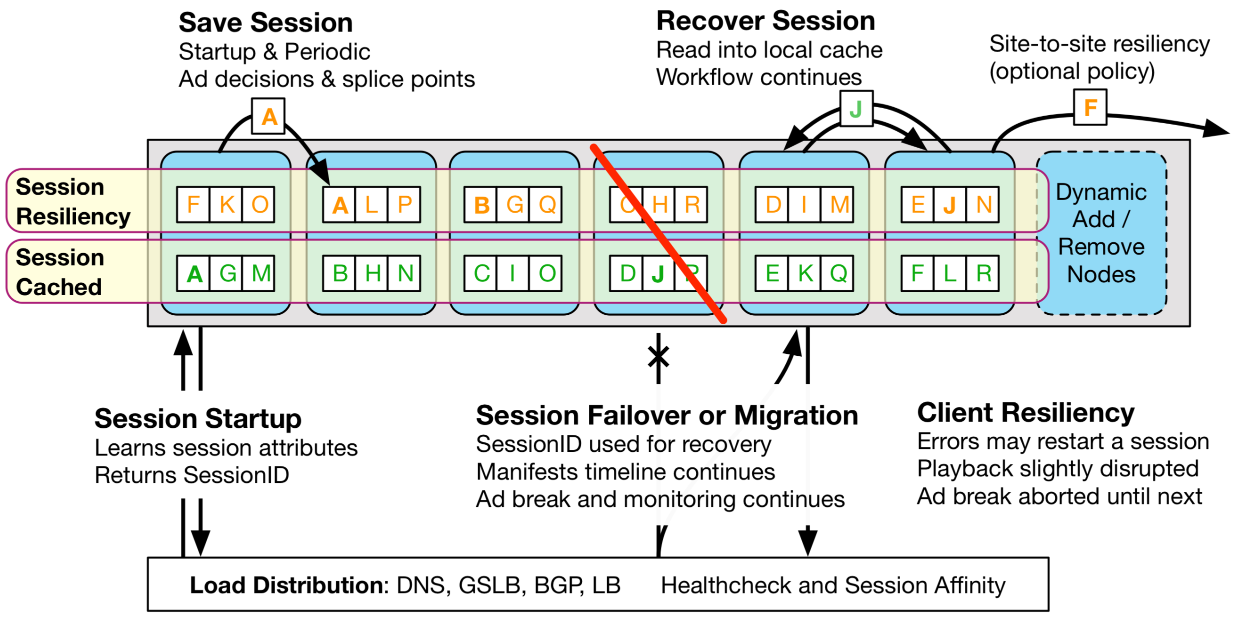
Save Session
Startup & Periodic
Ad decisions & splice points
Recover Session
Read into local cache
Workflow continues
Site-to-site resiliency
(optional policy)
A
J
F
Session Resiliency
Session Cached
F K O
A L P
B G Q
C H R
D I M
E J N
A G M
B H N
C I O
D J R
E K Q
F L R
Dynamic Add / Remove Nodes
Session Startup
Learns session attributes
Returns SessionID
Session Failover or Migration
SessionID used for recovery
Manifests timeline continues
Ad break and monitoring continues
Client Resiliency
Errors may restart a session
Playback slightly disrupted
Ad break aborted until next
Load Distribution: DNS, GSLB, BGP, LB
Healthcheck and Session Affinity

Figure 5 - Distributed session state caching model

With service availability, a device should be able to retain access to the service if one of the sites becomes unresponsive. This can be achieved without requiring session state replication across node sets, thereby saving network costs.

With service resiliency, a session should continue uninterrupted should one site become impaired or unresponsive, and the load be redirected to the other site. This can be achieved by setting replication policies ensuring each session state will be distributed to both node sets. However, this level of resiliency requires the appropriate network bandwidth to be provisioned between sites.

In the horizontally scaling design favored by Charter and depicted in Figure 4, session state replication can be thought of as a multi-level recovery mechanism that optimizes resiliency, service availability and performance:

- L1 – The individual node internally manages active sessions
- L2 – Automatic updates to neighboring nodes in the same site facilitate distributed resiliency within the site
- L3 – Distributed resiliency to nodes in a separate site can be optionally configured

The session state replication mechanisms are self-contained within the node logic, which removes any dependencies and bottleneck issues on a would-be central session state manager. In fact, session replication is independent of the management layer used to provision services on the nodes. Furthermore, a session is not dependent on other nodes, as replication is updated in the background without locking any real-time processing service. Replication is updated at decision points (ad breaks, bit rate switches in live HLS streams, etc.) and mid-window to maintain timeline consistency.

This design enables linear scaling of resources relative to the workload without any limitations. Let S be the number of sessions each node can handle, and N the number of nodes. Then:

- The memory in each node holds 2*S sessions, i.e. S+(N-1)*S/(N-1)
- The network interface at each node handles traffic for replica transmission of S sessions and for ingest of replicas for (N-1)*S/(N-1) sessions, totaling 2*S sessions
- The network interconnect backbone needs to handle the total load of a site's replication traffic, or S*N
- The option to configure extra cross-site replication doubles traffic requirements, i.e. 2*S*N

The two graphs below show the activity recorded during a real-life test that was performed on a 150K-session load served by two sets of load-sharing manifest manipulator nodes, with each set provisioned to handle the full load. At 16:37, one of the two systems went down, causing the sessions to seamlessly move to the other systems without interruption.

Observe that the volume of manifest requests (Figure 6) and peak sessions (Figure 7) went up on the surviving system in the same proportion as the activity went down on the other system. Soon after the down system was put back in service, this action restored an evenly distributed load across systems. At 16:52, the same procedure was repeated by reversing the roles of the systems. Throughout this process, we also measured non-HTTP 200 messages and logged none, further evidence that the clients were unperturbed throughout the whole test.

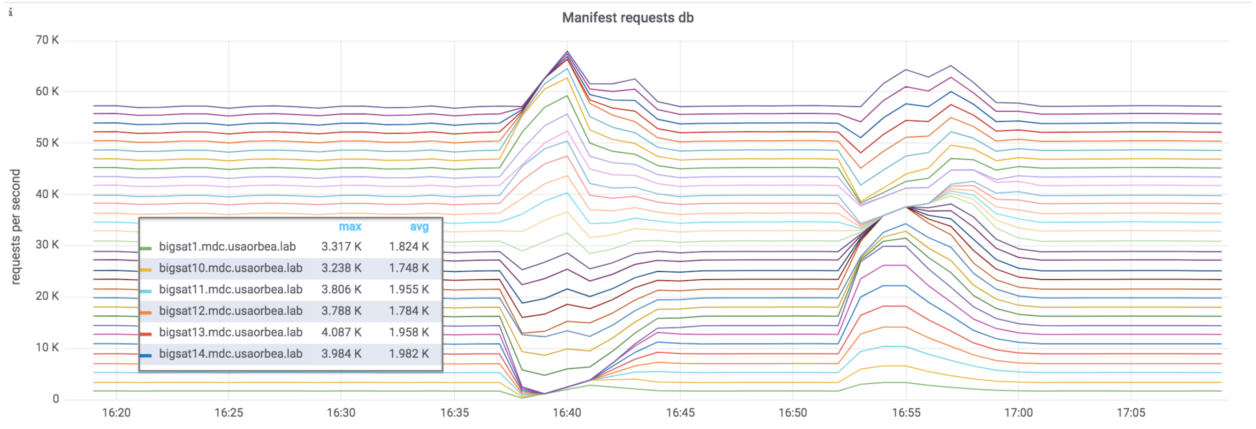

Figure 6 - Manifest requests on two load-sharing sets of manifest manipulators

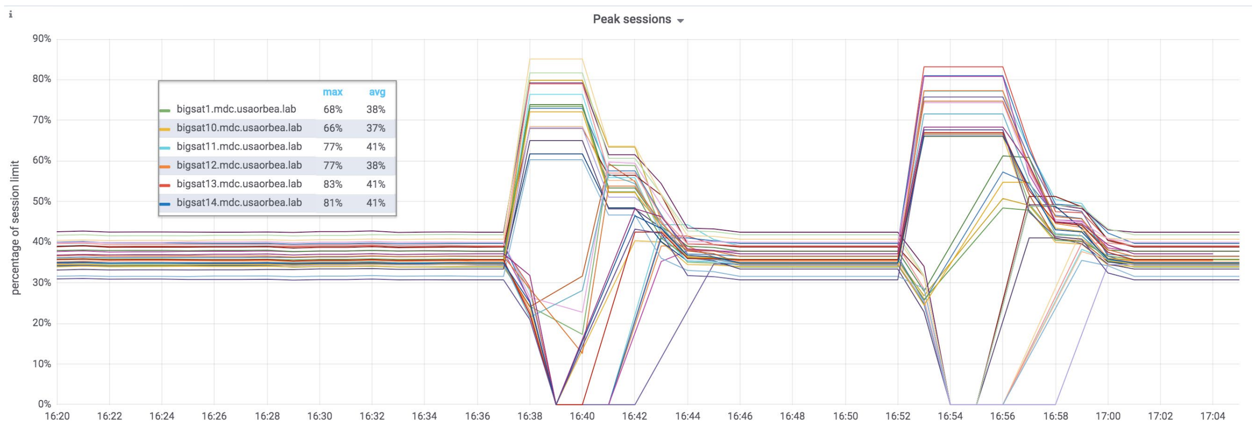

Figure 7 - Peak streaming sessions on two load-sharing sets of manifest manipulators

The scaling model presented in this section meets all the requirements of a modern software application that must support workload fluctuations, service availability and resiliency, and the ability to maintain/upgrade the system with agility and reliability.

This architecture provides Charter more flexibility in growing capacity either centrally for more efficient use of compute resources or at the edge, thereby reducing load on the network and optimizing client response times. As is often the case in solving complex problems, this is not an either or proposition, which makes the flexibility of horizontal scaling very attractive.

6.2. Load distribution

Coming up with a well-designed, load balancing system to address the specific needs of ABR video relies on a combination of widely used best practices and optimization techniques.

Equal load distribution is the goal. A new session can be handled at any acceptable node (note: different pools of nodes may be configured to handle different services, such as VOD vs live).

Load balancing should favor the "stickiness" of a session to the resource initially assigned to serve it. That applies to the client requests for manifest updates throughout the session, as well as to client requests for callback and tracking signaling (end-of-session, tracking events when proxied by the manifest manipulator). All this is most easily achieved using a load balancer that distributes load based on client IP address.

Client-IP-address-based distribution has many benefits, including:

1. Greater cache efficiency and lower latency for the session on-going updates (avoids the delay/cost of a session restore)
2. Ability to leverage caching of prior backend manifests being actively spliced into the output
3. Improved diagnostics for a client support issue, because it obviates the need to scan for IP addresses across the system
4. Protection of the client from any issues elsewhere in system (fewer system dependencies)

Client identifiers such as a key or hash can also be used. This method better supports the scenario of clients whose IP address may change during a session, as is the case with mobile clients that may switch between Wi-Fi and cellular networks, as well as clients behind NAT, DHCP, or VPNs.

Load may also be directed by systems, specifically:

- Content Services – Client-provided playback URL with hostname of a site (household region or other attributes)
- Resolution Services – Regional systems such as DNS/GSLB, DHCP, session router/gateway, select the site
- Network – Route to the site with the lowest cost border gateway protocol (BGP), fewest hops, lowest latency, peering bandwidth costs
- Balance – Spread sessions across servers or sites and adapt to available capacity, response times, and resiliency

Or by partitioning services, such as:

- Content Type – Directed to a specific cluster, access point, workflow (premium channels, VOD, PPV)
- Account Attribute – Class of service or assigned region for the subscriber/household/user info
- Device Tag – Provision or assign an endpoint to a regional access point or class of service

Load balancing factors as a key dependency if capacity expansions are realized either by proactive provisioning or elastic expansions. Proactive provisioning is the idea of adding servers or sites for anticipated larger loads without the need to reclaim the resources − which is appropriate when the use of the extra servers can be justified as an investment in anticipation of more customer acquisitions or organic usage growth. Elastic expansion is about quickly deploying more servers or sites for punctual needs, knowing the extra capacity will be reclaimed after the large-volume streaming event.

Finally, the load balancer plays a key role in system resiliency at many levels:

- Manifest manipulator node – The load balancer monitors health check for status / capacity to distribute sessions across nodes
- IP Fail-over – The load balancer's heartbeat fail-over (VIP) or site route health inject (RHI / BGP) mechanism can quickly reroute traffic
- Overflow – A safety valve mechanism that helps protect users' streaming experience by automatically bypassing the manifest manipulator when nodes are saturated or the manifest manipulator is inaccessible
- Network – Networking issues from client (home, service-group) and into the deployed site
- Client – Recovery attempts for retries, reconnects, restarts, user behavior, network roaming

Initially, Charter designed a content-based load balancing scheme for the manifest manipulation function, wherein sessions would be routed to siloed pools of resources based on the channel being requested by the client. At smaller scale, it was relatively simple to implement. As streaming activity grew, load grew more uneven across the siloed pools as the systems serving the more popular channels were taxed more often and sometimes saturated while capacity was unused on other systems.

To scale, Charter borrowed from some of the techniques we just reviewed. It devised a flat load balancing scheme combined with BGP routing that established reachability of the entire pool of manifest manipulation nodes and evened out the distribution of sessions across those resources. That system also allowed more flexibility in the ability to influence direction of load and traffic. For instance, in order to minimize traffic on the core network, Charter had the ability to move its manifest manipulation resources to the edge and tune its routes such that clients would enter the "closest" edge and be served by the least busy resource at that edge.

Charter also takes care to plan for the adverse impact of "roaming" clients that may change networks and thus may be routed to geographically separate manifest manipulation resources throughout a session. These issues are overcome by leveraging hashing mechanisms using network-invariant parameters associated to the client, combined with more robust session recovery mechanisms between sparse manifest manipulation resources (see Figure 8).

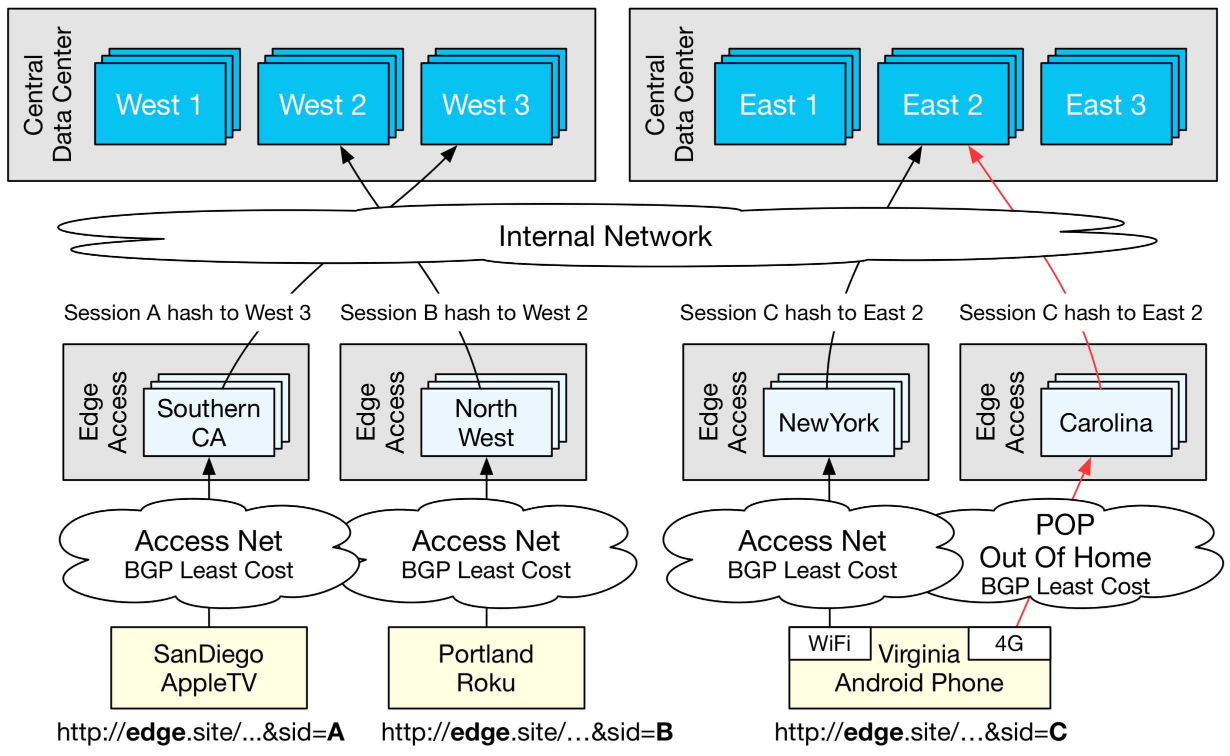

Figure 8 - Load distribution model in a multi-region deployment

6.3. Appliance to Data Center transition: the importance of hardware tuning

Over the years, Charter has gathered real experience and knowledge in how to leverage hardware virtualization technology to adapt its platform management policies to high performance application needs. An important aspect of manifest manipulation is the management of all the latencies in the critical path to delivering the stream to large volumes of clients.

Virtualization offers hardware independence and enables leveraging continuous improvements in processor, disk, and memory speed and size to offset cost of growth. However, Charter has found that tight policies are required in how the virtual platform is managed to avoid sacrificing performance and service quality.

Manifest manipulation is a mission critical application that directly impacts the user's quality of experience (QoE). When it works well, the application masks a lot of the complexities and interactions that happen behind the scenes to deliver a stream uniquely tailored to each connected client. Performance optimization is about ensuring that the application has uninterrupted and reliable access to compute, memory, and storage resources to minimize the delays of processing. The application tolerates latencies with Packager and ADS services within certain limits by design.

Charter evolved its deployment over the years. It started by deploying manifest manipulation on bare metal. As the scale of the deployment quickly grew, Charter moved the application to a virtualized infrastructure to provide better economy of scale and streamline the management and operation overhead of the platform. It took a sustained effort to tune the performance of the application running on top of a virtual environment. The following paragraphs describe the lessons learned along the way.

The initial toolset and virtual stack employed were still maturing to handle video streaming workloads. Tools like tcpdump were required to trace frequent lost SYN packets to the virtual switch. In addition, the system had to be further partitioned and the TCP stack was adjusted to more quickly abort the connection and retry to fetch the manifest from other CDN edges. The manifest manipulator was enhanced to include manifest caching, which greatly reduced downloads from CDN edges and relaxed the application's susceptibility to download latency, while preserving timely responses to client requests.

CPU contention was another challenge as other VMs sharing the host would briefly stall the throughput and responsiveness. While reports showed average resource contention below maximum capacity, they masked punctual spikes of usage by other applications sharing the host, thus starving the manifest manipulor of CPU cycles long enough to cause the potential for bitrate downshifting, which means degradation of the streaming user experience. This was detected using Linux metrics for CPU-stealing and internal health pings that showed gaps in CPU execution.

Charter even changed virtual stacks in the process of growing its infrastructure. The lessons learned from the tuning efforts described above proved valuable in reducing the cost of this transition. However, with more applications running on VMs, other issues needed attention.

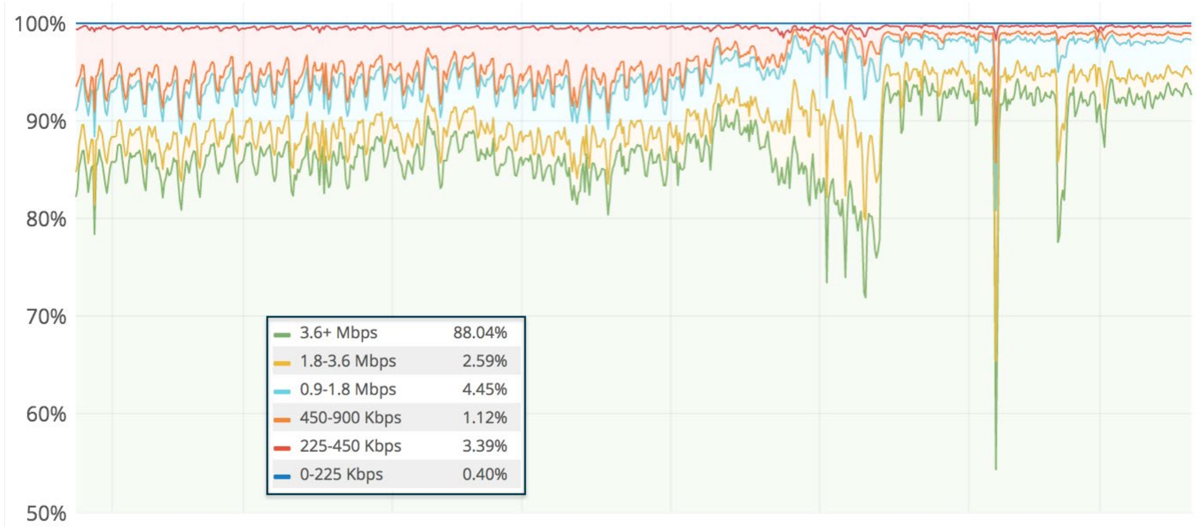

Figure 9 - Percent of client requests at different bitrate qualities

The graph in Figure 9 plots the percentile of each bitrate in the Charter ladder relative to all client requests recorded over a two-month period after the migration. It shows that clients had a low error rate, but quality levels being played were lower than expected.

This was primarily resolved by increasing capacity at the CDN edges to improve download rates for media segments, and by tuning routing to reduce delay of manifest fetches from the CDN. After these improvements, the average user playback quality was significantly improved. The remaining lower quality levels may be accounted for by mobile clients on Wi-Fi or roaming out of network.

Reviewing the metrics developed as part of the first virtual stack roll-out also helped quickly detect instances of CPU execution gaps on the second virtual stack deployment. The reported metrics allowed teams from video operations, infrastructure, and application vendors to work closely together and isolate cases of automated migration of manifest manipulation instances between hosts. The migration caused a stall with the bit rate downshifting effect previously described. Despite the fact that automated VM migration is common practice in data centers, this case helped shed light on the fact there is no cookie-cutter approach to managing applications in a large scale IP video deployment. Automated migration was ultimately disabled for manifest manipulation.

6.4. Multi-ADS support

Another scaling dimension that may be less obvious is measured in the platform's ability to accommodate new ad sales models and relationships. Service providers may open part of their own ad avails to third parties.

The challenge is that opportunities of this type often require working with external and third-party Ad Decision Systems because the parties involved want to avoid having to integrate their backend sales order, ad campaign, and ad preparation workflows with the service provider's own Ad Decision System.

At Charter, the manifest manipulator is equipped with the ability to share ad breaks among multiple Ad Decision Systems, thanks to its ability to route based on ad marker policies, on channel or content, or on parameters conveyed in client URLs. Further support of VAST Wrappers means that Charter retains full control of how an ad request gets fulfilled, because the request first goes to Charter's Ad Decision Sytem for fulfillment of its own inventory before it redirects the manifest manipulator to other third-party ADSs to request ad decisions to fill the remainder of the ad break.

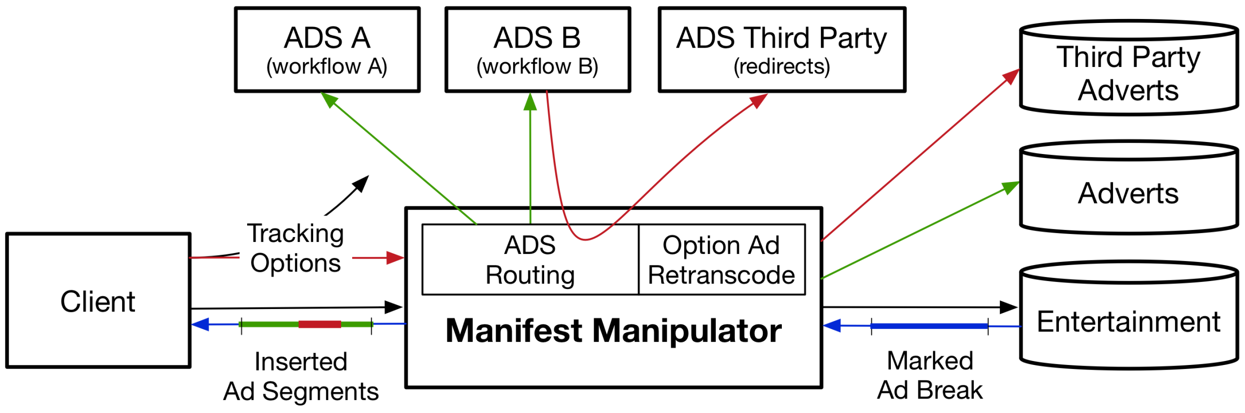

Figure 10 - Manifest manipulator routing to multiple Ad Decision Systems

Note the presence of a Retranscode function that serves to automate the adaptation of third-party ads into the operator-specified ABR profiles.

It is then up to the manifest manipulator to properly convey to the device client the tracking events specified by the various ADSs so that ad measurement and verification flow as expected to each party to the transaction.

Charter engineered its platform to be flexible and expandable to support not just its own avails, but also to enable more creative sales models providing win-win opportunities with other companies in the Television Advertising industry.

6.5. Client diversity and integration

At Charter, the manifest manipulation platform is designed to support DAI and alternate content switching for both HLS and DASH streaming clients. While these two protocols share the same core concept, there are fundamental techonology differences. For HLS, the manifest manipulation occurs at the sub-level manifest and involves individual video and audio profile manifest requests. The manifest manipulation in DASH occurs at a single manifest level which encompasses the adaptation set for various video and audio profiles.

In HLS streams, the primary audio stream for the content is most often packaged in the same media segment as the video stream. Some content may involve multiple language versions of the audio experience (e.g. multiple audio streams), which are packaged in secondary tracks fetched separately from the video segments by the client. These multiple manifest requests can drive additional load on the manifest manipulation and should be taken into consideration while designing for scale.

The other key consideration concerns any variation in encoding/packaging profiles between the entertainment content stream and the ad or alternate content stream. As the ads and entertainment streams originate through different sources and the encoding/packaging is managed and operated by different operations groups, there are bound to be variations in how the bitrates, resolutions, and other parameters are specified in the manifest. The manifest manipulation application needs to be robust enough to handle these variations without causing any adverse behavior in the client player.

6.6. Metrics and Monitoring

In order to assure high SLAs and achieve high availablilty, the manifest manipulation platform plays a key role by providing the hooks and the data to collect all service level and performance metrics and build scalable monitoring. The data provided by the platform address both overall performance of video sessions and performance for ad decisioning and insertion.

Key monitoring metrics include:

- Success rate/Errors for all client requests
- Success rate/Errors for all interactions with the ADS
- Latencies from CDN and ad request processing
- Any CDN errors for fetching the content and ad manifests
- Rogue requests to the application
- Shifts in the bitrate manifests

They help build a robust analytics dashboard to monitor:

- Overall performance of the video streaming platform and any adverse impact to user experience
- Overall performance of the ad placement platform, including measurement of missed monetization opportunities
- Performance of a specific ad campaign or ad creative and success with the placement

Leveraging the telemetry data and log infrastructure instrumented in its manifest manipulation platform, Charter was able to build tools tailored to the specific needs of the Video Operations and Ad Operations teams to enable shared and complementary monitoring of system performance and of individual application process and functions.

Charter leveraged the following dataset from the application:

1) HTTP stats with endpoint URLs for all northbound, soutbound and ADS requests
2) Logs for all ADS requests and responses
3) Errors from manfiest processing, including under-runs, over-runs, and request timeouts
4) Aggregate metrics for ads placed in particular content streams
5) Logs for processing time to estimate minimum, maximum and average latencies
6) Logs for individual client session context
7) Logs for filtering of rogue requests

By harnessing the granularity and depth of the manifest manipulation data, coupled with datasets from other points of the delivery pipeline, it is also possible to more effectively mitigate and resolve complex issues.

Conclusion

Charter learned valuable lessons and drew major benefits as a result of implementing changes to scale its IP video and advertising system.

Leveraging horizontal scaling for manifest manipulation significantly improved the operator's ability to cost-effectively expand the system on its own infrastructure. As a result, Charter can better adjust to growing loads and deploy temporary instances on cloud infrastructure to absorb occasionally large load spikes.

By taking advantage of the large arsenal of standard routing and load balancing techniques, Charter was able to achieve more efficiency through even distribution of the load.

To scale with more agility, Charter moved manifest manipulation from dedicated appliances to virtualized infrastructure, establishing clear operational policies to ensure the constant availability of compute resources and ensure high quality session handling.

Technological innovations leading to the support of multi-ADS routing helped expand Charter's monetization opportunities.

Leveraging manifest manipulation has helped Charter speed up new client integrations while adapting the manifest delivery to ensure a user experience consistent across an ever-expanding landscape of IP-connected devices.

Finally, Charter leveraged its experience with IP video delivery to define and implement a greater breadth of Quality of Experience metrics and enhanced its data infrastructure in order to more effectively monitor its network and applications at scale.

Today, Charter serves IP video streams in large and growing volumes. With a more scalable architecture and the operational tools to grow, Charter sees few challenges ahead on its path to meeting or exceeding broadcast quality delivery for IP Video. Standardization efforts anchored by the Common Media

Application Format (CMAF) will go a long way to improving scale and reducing content management complexity by allowing common media and encryption across HLS and DASH platforms and many native DRM systems, just as chunked-based encoding and transfer will help reduce ABR latency.

Charter did not make lightly the decision to invest in IP, and for all the efforts made to get this far, there is still the major challenge many operators face to figure out how to transition to the future of IP. The question now is, "How to leverage the investments made in the IP video workflows, a robust and scalable CDN backbone, and advanced Advertising Management Systems, to effectively serve and monetize both IP and QAM platforms over what can be predicted to be a long transition period?"

Abbreviations

ABR	Adaptive Bit Rate
ADS	Ad Decision Service
ACDS	Alternate Content Decision Service
BGP	Border Gateway Protocol
CDN	Content Delivery Network
CMAF	Common Media Application Format
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DRM	Digital Rights Management
GSLB	Global Server Load Balancing
HLS	Hypertext Transfer Protocol Live Streaming
HTTP	Hypertext Transfer Protocol
HTTP-MPEG 2 TS	Hypertext Transfer Protocol – MPEG 2 Transport Stream
IP	Internet Protocol
ISBE	International Society of Broadband Experts
LB	Load Balancing
MHz	megahertz
MPEG-DASH	MPEG – Dynamic Adaptive Streaming over Hypertext Transfer Protocol
MPEG TS	MPEG Transport Stream
NAT	Network address translation
OTT	Over the Top (video service)
PPV	Pay per view
QAM	Quadrature amplitude modulation
QoE	Quality of Experience
RGUs	Revenue generating units
(RHI / BGP)	Route health injection / Border Gateway Protocol
SCTE	Society of Cable Telecommunications Engineers
SLA	Service Level Agreement
SSAI	Server-Side Ad Insertion
STB	Set-top box
SYN	Synchronize
TCP/IP	Transmission Control Protocol/Internet Protocol

TTL	Time-to-Live
UDP	User Datagram Protocol
VAST	Video Ad Serving Template
VIP	Virtual IP
VM	Virtual Machine
VOD	Video-on-Demand
VPN	Virtual Private Network
Wi-Fi	Wireless Fidelity (IEEE 802.11X)

Bibliography & References

[1] Strategy Analytics Global Advertising Forecast – 2010 -2030, Michael Goodman, August 30, 2018

[2] *Q1 2019 Conviva's State of the Streaming TV Industry*; Conviva

[3] Extreme Reach Q3 2018 Video Benchmarks, Mary Vestewig, October 31, 2018

[4] Industry Voices—Dan Rayburn: Handle Manifest Manipulation at the network edge to personalize video experiences; Fierce Video, Jul 1, 2019