Intelligent Live Origin and Resilient Streaming

Xiaomei Liu
Netflix Inc.
California, USA
xliu@netflix.com

Rajasekhar Ummadisetty
formerly Netflix Inc.
Washington, USA
kanna14243@gmail.com

Joseph Lynch
Netflix Inc.
Maryland, USA
josephl@netflix.com

Chris Newton
Netflix Inc.
Texas, USA
cnewton@netflix.com

ABSTRACT

Live streaming resiliency is critical due to the high impact of failures. Current DASH redundant encoding and packaging for segmented live media architectures use redundant paths and epoch-based synchronization for interchangeable segments across cloud pipelines. However, this approach faces challenges, including segment template timing mismatch issues, distribution of defective segments, content misalignment causing playback glitches, and inefficient scaling of multi-publishing redundancy. This paper introduces solutions leveraging segment template-aware origin servers to improve scalability and resiliency. By utilizing predictable segment schedules and multi-pipeline awareness, the origin can enhance CDN operations, select non-defective segments, and employ pipeline prioritization to minimize audio-visual glitches. Furthermore, we propose scaling redundancy without multi-publishing by using multi-region distributed storage and asynchronous cross-region propagation, offering a more scalable and decoupled architecture.

CCS CONCEPTS

• **Information systems → Multimedia streaming**

KEYWORDS

Resiliency, Redundancy, Streaming pipeline, Multi-region, Live origin, Cloud Storage, CDN

ACM Reference format:

Xiaomei Liu, Rajasekhar Ummadisetty, Joseph Lynch, Chris Newton. 2026. Intelligent Live Origin and Reliable Streaming. In Mile-High Video Conference (MHV '26), February 2–5,2026, Denver, CO, USA. *ACM, New York, NY, USA, 7 pages.* https://doi.org/10.1145/3789239.3793275

1 Introduction

Live streaming for entertainment and sports continues to thrive with increasing viewership, solidifying its position as a mainstream consumption model, driven by evolving viewing habits that favor flexibility and interactivity as well as the streaming platform's capability of expanding content reach beyond traditional broadcast systems.

Live streaming is based on adaptive streaming with segmented content. The distribution processing involves multiple stages from the distribution encoder and packager to the origin, before content is distributed via CDN to reach end users. For better scalability, this multi-stage distribution processing, which is called encoding pipeline, is typically implemented in the cloud. Resiliency is normally improved with added redundancy throughout the system, from contribution to distribution processing.

The reference architecture of redundant encoding pipelines is specified in DASH Redundant encoding and packaging for segmented live media (REaP) [1] and is illustrated in Figure 1. The redundancy works by components from one stage sending outputs to the components of the next processing stage. When a component fails in a stage, the components in the next stage still have a valid input, eliminating the single point of failure in the pipeline. Pipeline is often deployed in different cloud regions to avoid regional infrastructure failures.

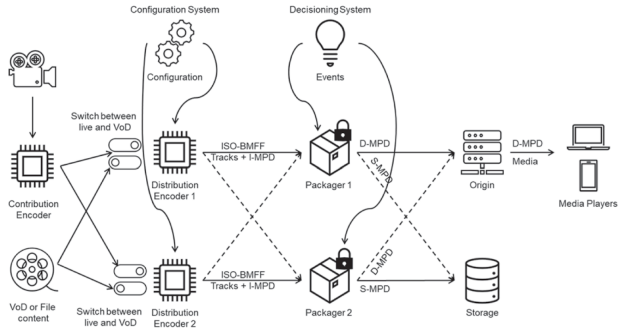

Figure 1 — Reference architecture of redundant encoding and packaging of live segmented media[1]

Early work was done by Unified Streaming [2] on encoder and pipeline synchronization without direct communication among encoders. The design used a common time anchor to create interchangeable segments by the encoders. When segments are interchangeable, a processing component can select input from any upstream components without interrupting downstream consumption. The solution is further standardized by the ISO/IEC FDIS 23009-9 [1], specifying the usage of constant segment duration to facilitate segment creation and embedded timecode in the contribution feed to convey the common time anchor, such as UNIX epoch, to the encoder.

The epoch-based, multi-cloud-region pipeline architecture presents challenges. Although constant segment duration enables clients to use a segment template without manifest refresh, client/server clock timing mismatches with the template can cause scalability

MHV '26, Denver, CO, USA

ACM ISBN 979-8-4007-2466-4/2026/02
https://doi.org/10.1145/3789239.3793275

issues in large-scale deployments. In addition, while the multi-pipeline interchangeable segments assumption works well when these segments are not defective, this assumption can cause a defective segment to be served when the segment from one pipeline is defective and the segment from another pipeline is not. Moreover, segment content alignment is often not guaranteed despite segment functional equivalence. This misalignment is often a source of audio-visual glitches at playback due to mixing segments from different pipelines at the segment boundary. Lastly, the DASH REaP architecture's reliance on multi-publishing from the encoder and packager to the next pipeline stage for redundancy doesn't scale efficiently beyond a dual pipeline setup.

This paper introduces solutions to enhance the scalability and resiliency of the redundant cloud encoding pipeline architecture by incorporating intelligent live origin components. Section 2 describes the segment-template-aware origin and how it improves live streaming distribution. Section 3 introduces the multi-pipeline and multi-region aware origin, incorporating media awareness to select better pipeline candidates and enhancing the streaming resiliency with pipeline priority and candidate selection consistency. Section 4 presents a scalable multi-region redundancy architecture with storage layer cross-region propagation.

2 Segment-template Aware Origin

The constant duration segment template brings predictability to segment publishing. This publishing schedule intelligence enables the origin to enhance the CDN streaming and scalability. With constant segment duration, streaming clients can predict the live edge segment without periodic manifest refresh [3], reducing the burden both on clients and the CDN. The success of the segment template (i.e, start segment number, segment duration, availability start time) relies on client and server clock synchronization. With large-scale streaming and tens of millions of clients, not all clients have always synchronized timing with the server, even with the best clock synchronization algorithm. This often causes clients to fetch segments ahead of publishing time, resulting in HTTP 404 (Not Found) errors [4].

Situated at the end of the cloud encoding pipeline, the origin is aware of the segment publishing schedule and can easily predict the expected publishing time of a requested segment. This allows the origin to intelligently field requests for segments that arrive early - when the origin determines that a 404 is to be returned, an appropriate TTL (e.g., via a Cache-Control: max-age) can be added so that the CDN will hold that response until the segment is expected to be published. In addition, the origin server can provide a 'hold open' metaphor [5] for the next segment, so rather than returning a 404 for the next segment of an active stream, the origin server will hold the request until the file is available at which time it will return it. These options significantly reduce unnecessary chatter within the network. By providing the segment template to the CDN, it is also possible to block requests for incorrect requests; for instance, for segments that are beyond a time threshold from the live edge so that there is no need to even contact the origin server.

The publishing awareness also enables the origin to provide valuable insights on client/CDN fetching, facilitating continued improvement in the overall system. Figure 2 shows sample error metrics on the live origin, highlighting client/CDN fetch errors:

- The FetchEarlyException indicated that clients/CDN were fetching before the expected segment publishing time.
- The GoneException showed that the segment fetch exceeded the segment generation deadline in live streaming.
- The NotFoundException indicated that the segments fetched were not available, but may potentially be generated later, i.e. the segment generation deadline had not passed.

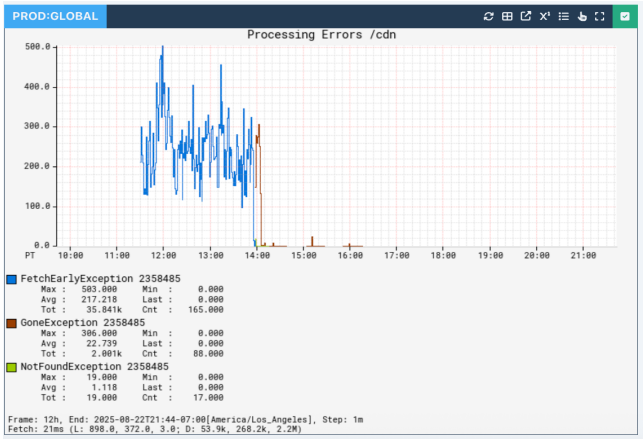

Figure 2. Example origin metrics on CDN fetching

3. Multi-region and Multi-pipeline Aware Origin

The multi-region and multi-pipeline redundancy design increases the resiliency of live streaming. In the redundancy reference architecture shown in Figure 1, if one pipeline or component fails to produce a segment, the multi-pipeline-aware origin can select the corresponding segment from an alternative pipeline. Having the failover handled at the server side, transparent to streaming clients, improves streaming resilience.

3.1 Media-aware Resiliency Enhancement

Although the above segment selection works well if one pipeline completely fails to generate the segment, it does not work well when corresponding segments from the dual pipeline are both generated, but one has defects, due to the assumption of interchangeable segments in the reference architecture. With the multi-region and multi-pipeline design, pipelines are typically independent, and the chance of concurrent defects is low. If there is an indication of segment defects, the origin can select the best candidates instead of treating segments as interchangeable. The origin iterates through segment candidates, from high-priority to low-priority pipelines, and identifies the first non-defective candidate to serve. If all candidates are defective, the high-priority pipeline candidate is used. The origin is expected to have some knowledge (e.g. via configuration): pipeline priorities, segment duration, and URI scheme to identify segments.

HTTP headers can be used to deliver additional metadata. Defects such as mismatched sample counts, TFDT timing, and segment numbers can be detected at the packager based on ISO/IEC 14496-12 fMP4 [6]. The defects can also be detected by the encoder. In DASH-IF Live Media Ingest Protocol [7], segments with black frames/silent audio/empty timed text are labeled with SegmentTypeBox ("styp") *slat* brand. The defects can be signaled to the origin via custom HTTP headers such as: "timing-discontinuity:true", "sample-count:60", "slate:true". Similarly, the pipeline identifier can be signaled: "pipeline:us-east-1". Although there is currently no standardization of these custom HTTP headers, industry forums such as DASH-IF can standardize the headers and enable interoperability.

If streaming systems use manifests with explicit segment listings, such as HLS [8], the pipeline-aware origin can improve resiliency by consolidating manifests from different pipelines. For example, when there are disruptions to pipelines at non-overlapping times during a live event, the consolidated manifests avoid gaps in the segment listings.

3.2 Pipeline Synchronization and Prioritization

The redundancy design of ISO/IEC FDIS 23009-9 architecture is based on the segment interchangeability assumption, which may not be alwyas achievable. The modern contribution processing air chain has many processing stages, including audio/video pre-processing, graphic overlay, ad marker insertion, and compliance check, among others. Contribution redundancy utilizes independent air chains to prevent a single point of failure. To make the segment interchangeable, the content and the associated anchor time from the redundancy contribution feeds must align. The complicated contribution processing and independent air chain often make segment interchangeability challenging. For seamless ad splicing, the ad marker and the video IDR frame at the splice point must be frame-aligned, making the contribution content alignment across pipelines even more difficult. When the segment interchangeability assumption is broken and redundant pipelines mix segments from different pipelines to serve streaming clients, the end user will experience audio-visual glitches, even if the client playback is without failure.

Pipeline prioritization at the origin is an enhancement that works when the encoding pipelines are NOT content synchronized. Both pipelines are still maintaining timing alignment, meaning that the TFDT timestamps for segment(N) from both pipelines are the same, despite the content in the segments mismatching. The regional pipelines have a priority associated with them. When the live origin selects a segment from the pipelines, it prefers the prioritized pipeline/region. The live origin is applying a configurable jitter guard to wait for the cross-region segment publishing to finish before making the pipeline selection. With awareness of the publishing schedule, the origin adds a jitter guard of O(sec) to the expected segment publishing time before candidate selection.

The live origin only selects segments from the lower-priority pipeline when segments from the higher-priority pipeline are not available or have defects. The jitter guards enable pipeline stickiness (i.e., consistently selecting the high-priority pipeline under normal conditions) at the live origin. When a pipeline fails (not generating segments for a period), the origin automatically switches over to the alternative pipeline. Once the failed pipeline recovers, the system returns to its original state of multi-pipeline redundancy. There may be a visual glitch during the pipeline switchover if the pipeline content is not properly aligned. Given pipeline failure is rare, this design ensures both consistency and resiliency without the strict requirement of pipeline alignment. In contrast, without pipeline priority and stickiness, switching pipelines at the segment boundary can introduce frequent, random visual glitches, depending on when CDN/client requests arrive and when the corresponding segments become available. This negative impact is not only limited to live edge access, but also affects DVR playback, given the caching effect of the CDN.

Figure 3 illustrates the pipeline selection stickiness in an example live event streaming with priority encoding pipeline (ENC1) in the AWS us-east-1 region and an alternative encoding pipeline (ENC2) in the AWS us-west-2 region. The setup included a 2-second segment duration with a maximum bitrate of 12 Mbps CBR and a 3-second jitter guard. The encoders with the endpoint "origin" are publishing encoders/pipelines, while encoders with the endpoint "cdn" reflect which encoders/pipelines are served to the CDN. The Y-axis represents the number of HTTP requests to the live origin during a 1-minute interval. It clearly demonstrates the consistency of pipeline selection: only the priority pipeline (enc1) was served during the 3-hour (16:30 - 19:30) live event streaming, while both pipelines were active. It is worth noting that pipeline stickiness does not prevent the live origin from selecting an alternative pipeline if an unusual circumstance arises where the priority pipeline fails to deliver timely segments or contains faulty segments. If a catastrophic pipeline failure lasts for a period of time, the content misalignment introduced glitch is limited only to the time of the pipeline switchover.

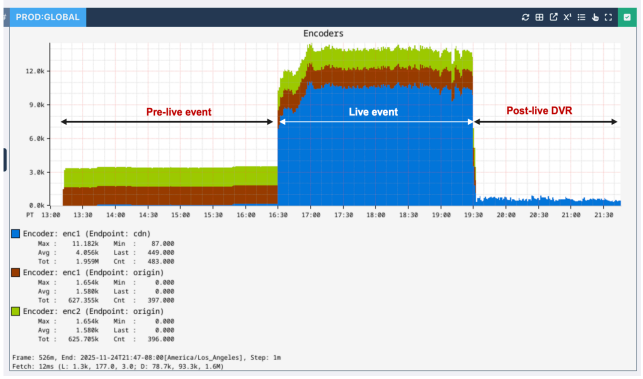

Figure 3. Pipeline stickiness at the live origin

There are tradeoffs between pipeline synchronization and distribution latency. If a particular implementation can ensure tight

content alignment among pipelines, then the need for a large jitter guard and pipeline stickiness decreases. Reducing the jitter guard can help reduce end-to-end latency without negatively impacting the end-user experience. Though O(sec) jitter guard at the origin is acceptable for live streaming, the resiliency architecture for ultra-low latency live streaming, such as LL-DASH/LL-HLS, poses new challenges that require further research and carefully balanced system design.

3.3 Pipeline Selection Consistency and CDN Impact

Having each encoding pipeline share a common epoch and segment construction template ensures that clients can have a good playback experience even if they switch between encoding pipelines from one segment to the next. The consistency cannot generally be guaranteed at the byte level however, meaning that anything that could cause a player to receive partial segments from two or more encoders must be avoided. Read timeouts mid download may occur, so any recovery, especially involving failover to an alternate distribution node needs to ensure that the same encoder's content is used.

If feeding content into the decoding pipeline at the player whilst retrieving the object, switching to different encoders' segment requires unwinding the old content first. Having the distribution node fetch alternates consumes double (or more, depending on the number of pipelines) the fill bandwidth and storage, so is best avoided. Having an origin server that can maintain consistency allows a single copy to be propagated into the CDN and through to the clients

4 Scale Beyond the Dual Pipelines

The reference redundancy architecture shown in Figure 1 illustrates the dual publishing of a processing component (encoder, packager, etc). This dual publishing introduces additional complexity in the design of the processing component. The component not only needs to allocate resources for the dual publishing, but it also needs to deal with publishing failure properly. Though dual pipeline redundancy can provide reasonable resiliency for regular live event streaming, it may not be adequate for big bet events with very large-scale audiences. When the pipeline goes beyond the dual regions to three or four regions, then the multi-publishing overhead becomes an issue for critical components, such as encoders and packagers.

In [2], an example is shown to split the three input signals to feed three origins/packagers. This redundancy design has several disadvantages. First, the operation becomes more complicated in managing which signal goes to which origin. Secondly, not all origin has visibility to all input signals, reducing the effectiveness of three-input redundancy. Furthermore, since not all contribution sources are visible to all pipelines, it is not feasible to implement pipeline prioritization. Without pipeline prioritization, pipeline content synchronization becomes a hard requirement for the redundancy design.

4.1 Redundancy with Origin Storage Propagation

A new redundancy design based on origin with multi-region storage propagation is a better approach. As shown in Figure 4, there are three fully independent pipelines, one in each cloud region, and three regional origin service components. At the center of this design is a multi-region full-active storage behind the origin service. Each origin service writes to its own region. The storage layer is responsible for cross-region data propagation. Each regional origin can read data propagated by the storage layer from other regions. For example, an origin in us-east-1 is able to read storage data originated from the us-west-2 pipeline. In this simplified architecture, the origin in each region has visibility to streaming data originating from all regions. This enables the origin to select the priority pipeline even if it is not in the same region as the pipeline. This also removes the requirement for encoder and packager multi-region publishing.

This design requires a datastore which replicates data globally. Some datastores such as Apache Cassandra [9] support native full-active global replication - data can be written and read from multiple geographic locations simultaneously. Most datastores do not natively support full-active, but custom replication solutions can be built to enable global replication, as demonstrated by Facebook's Wormhole [10] or Netflix's EVCache and Write-Ahead-Log (WAL) replication systems [11][12]. In either case, the live streaming encoding pipeline can delegate the complexity of cross-region data propagation to data management systems such as Netflix's KeyValue Abstraction [13], which seamlessly handles replication across storage layers.

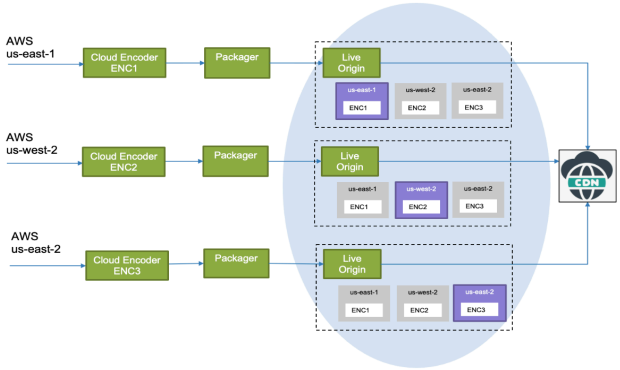

Figure 4. Multi-region redundancy architecture with storage propagation

The storage replication architecture allows but does not mandate full mesh replication. In Figure 4, each storage region under the live origin box is an independent storage namespace, where the replication topology can be configured as needed to meet CDN read requirements. Full mesh simplifies namespace management with a practically small number of pipelines and regions, but has cost implications in storage and network as the number of storage regions grows. In this architecture, each storage namespace can be configured to replicate to the most appropriate set of regions for a given pipeline, in coordination with the CDN top-tier.

Storage replication enables the removal of the tight coupling between the encoding pipelines and their origins. There is now the possibility of having more origin regions than the number of encoding pipelines. As shown in Figure 5, the two pipelines can have three or even four origin regions (or access points).

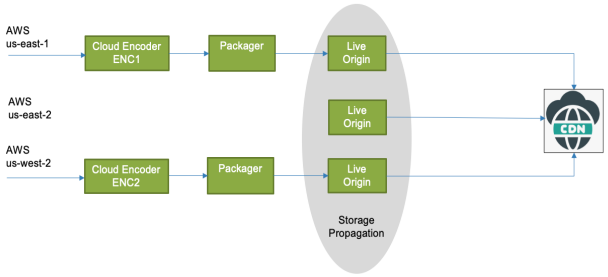

Figure 5. Flexible mapping of pipelines and origin access points

When the network is serving multiple simultaneous events, there is often diversity in which regional encoding pipelines are being used. Where the content is encoded should be determined by the best place to encode it. As the number of publishing points increases, the complexity of the hierarchy increases. What is preferred is to keep a single origin (and backup) for each CDN node, so that regardless of where it is being encoded, the CDN node will be able to fill content from the origin server best placed for it - a single hierarchy, with an optimal and consistent path to the origin, providing predictable network characteristics. Having a reliable distribution system at the origin layer that provides a flexible mapping of encoding pipelines to origin locations allows for configuration simplicity within the CDN, coupled with easier diagnostics and impairment analysis. Furthermore, link utilization is more predictable - there is a single hierarchy that can be more closely modelled on the underlying physical connectivity, rather than needing to support multiple overlapping event-based virtual hierarchies

4.2 Cross-region Propagation

The storage subsystem for the origin replicates across four geographic regions, replicating data globally via full-active *unordered* replication in Cassandra, as well as *ordered* replication through Apache Kafka which replays operations against remote Memcached instances (EVCache). This involves pushing hundreds of MiB with high reliability through Long Fat Networks (LFN) with up to 120 millisecond delay, illustrated by the simplified AWS network topology and RTT latency [14] in Figure 6.

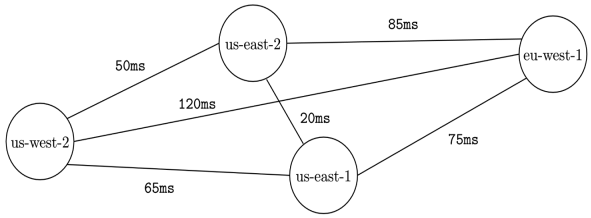

Figure 6: Network topology of AWS regions

To minimize latency of writes, the Netflix KeyValue service breaks video segments of variable size into 64KiB chunks [13] and sends those chunks in parallel to the durable store Apache Cassandra and caches them in EVCache. Chunks are acknowledged once a region-local quorum of Cassandra nodes have responded, but Cassandra behind the scenes is asynchronously replicating in an unordered fashion to remote regions. As Memcached does not implement replication, EVCache replication copies the chunk data via an ordered Kafka topic in a more traditional sequential approach. In production, we find unordered parallel replication consistently performs near ideal network latency with minimal (~10ms) additional delay, while ordered replication has significant (500ms+) delay as seen in Figure 7.

Example P50 and P99 replication latencies that power the visualization in Figure 7 are shown in Table 1:

Source	Destination	Unordered Replication (P50, P99) ms	Ordered Replication (P50, P99) ms
us-west-2	us-east-1	(88, 98)	(657, 1060)
us-west-2	us-east-2	(65, 73)	(615, 947)
us-west-2	eu-west-1	(152, 172)	(856, 1500)

Table 1: Replication latencies from US-WEST-2 region

Relying on unordered replication strategy leads to near optimal replication latency in the common case, but it has a tradeoff in that we may observe partially replicated data. Our design uses this as a feature, however, as even a single 64KiB chunk replicating is enough for remote readers to learn that another region has successfully stored a O(MiB) segment, meaning that independent WAN fallback in the CDN is possible as it has knowledge of which region produced the segment.

Figure 7: Observed cross-region replication latencies

4.3 Cross-region Network Reliability

Another important factor to consider is the reliability of the cross-region network. An unreliable cross-region network often manifests itself as occasional longer latency, large jitter, and failed or late responses. To increase the resiliency of this origin/storage-based propagation, several options or a combination of options can be applied:

- Increase the origin jitter guard to accommodate longer propagation delay. Larger jitter guards can have better pipeline selection consistency at the expense of end-to-end streaming latency.
- Limit the scope of cross-region propagation so the longest propagation delay is under control. For example, if the live streaming is distributed only in the US, then the cloud region used can be close, with cross-region latency better controlled
- CDN implements a fallback strategy among the supported origin regions to increase the success rate

In the fallback strategy, the origin differentiates between non-existent and late-propagate data. If the datastore propagates late, such as with partial data success, the origin directs the CDN to fall back to a different region's origin with a different error code from 404. Top-tier CDN nodes are configured to attempt multiple origin server nodes for certain error conditions, such as failure to connect or issue a request at the preferred location, while the content distribution system may still experience delays in publishing content to one origin region or another. As such, the top-tier CDN nodes should be able to fall back to alternate locations on definitive errors, such as a 404 or 500. These failover locations may be directly to the origin servers at a geographically distinct origin server location, or to the top-tier CDN nodes at such a location - although in that case, care is needed to prevent loops that could result in a deadly embrace.

5 Conclusion

As live streaming becomes the more dominant consumption path for entertainment and sports content, resiliency and scalability will always be important design areas alongside many of the other innovations in the live streaming system architecture.

The intelligent live origin plays an important role in facilitating simpler, more scalable, and more resilient redundancy architecture. Publishing awareness, media awareness, multi-pipeline awareness, and multi-region awareness all help optimize live streaming systems. The multi-region storage system, with its built-in cross-region propagation, provides many advantages not matched by the traditional multi-publishing redundancy architecture.

References

[1] ISO/IEC SC29 WG3. Text of ISO/IEC FDIS 23009-9 Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 9: Redundant encoding and packaging for segmented live media (REaP). 2024

[2] R. Mekuria, R. Ramos, J. Fletcher, M. Ogle, A. Wagenaar, D. Griffioen, B. van Dijk, and D. Andric. EpochSegmentSync: inter-encoder synchronisation for adaptive bit-rate streaming head-ends. In ACM MHV, 2022.

[3] S. Fedorov, C. Pham, F. Ribeiro, C. Newton, and W. Wei. Behind the Streams: Three Years Of Live at Netflix. Part 1. Netflix Tech Blog, July 15, 2025. https://netflixtechblog.com/behind-the-streams-live-at-netflix-part-1-d23f917c2f40

[4] Hypertext Transfer Protocol (HTTP) Status Code Registry https://www.iana.org/assignments/http-status-codes/http-status-codes.xhtml

[5] C. Newton. Techniques for selectively delaying responses to premature requests for encoded media content. October 17, 2024, Patent Publication No: 20240348848 https://ppubs.uspto.gov/api/pdf/downloadPdf/20240348848

[6] ISO/IEC 14496-12, Information technology — Coding of audio-visual objects — Part 12: ISO base media file format, 2022

[7] DASH-IF Live Media Ingest Protocol, Technical Specification, 28 February 2024 https://dashif.org/Ingest/

[8] R. Pantos RFC 8216 HTTP Live Streaming IETF RFC, August 2017, online: https://datatracker.ietf.org/doc/html/rfc8216

[9] Apache Cassandra documentation https://cassandra.apache.org/doc/latest/

[10] Sharma, Yogeshwer, et al. 2015. Wormhole: Reliable Pub-Sub to Support Geo-replicated Internet Services. 12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15). 2015.

[11] S. Madappa, V. Nguyen, S. Mansfield, S. Enugula, A. Pratt, F. Siddiqi. 2016. Caching for a Global Netflix, Netflix Tech Blog. (Mar 1, 2016). Retrieved from https://netflixtechblog.com/caching-for-a-global-netflix-7bcc457012f1

[12] P. Karumanchi, S. Fu, S. Rangarajan, V. Arvind, Y. Wang, J. Lu. 2025. Building a Resilient Data Platform with Write-Ahead Log at Netflix. Netflix Tech Blog. Retrieved from https://netflixtechblog.com/building-a-resilient-data-platform-with-write-ahead-log-at-netflix-127b6712359a

[13] V. Arvind, R. Ummadisetty, J. Lynch, V. Chella. 2024. Introducing Netflix's Key-Value Data Abstraction Layer. Netflix Tech Blog. Retrieved from https://netflixtechblog.com/introducing-netflixs-key-value-data-abstraction-layer-1ea8a0a11b30

[14] M. Adorjan, AWS Latency Monitoring, 2025. https://www.cloudping.co/