Stream: Internet Engineering Task Force (IETF)
RFC: 9317
Category: Informational
Published: October 2022
ISSN: 2070-1721
Authors: J. Holland, *Akamai Technologies, Inc.*; A. Begen, *Networked Media*; S. Dawkins, *Tencent America LLC*

RFC 9317
Operational Considerations for Streaming Media

Abstract

This document provides an overview of operational networking and transport protocol issues that pertain to the quality of experience (QoE) when streaming video and other high-bitrate media over the Internet.

This document explains the characteristics of streaming media delivery that have surprised network designers or transport experts who lack specific media expertise, since streaming media highlights key differences between common assumptions in existing networking practices and observations of media delivery issues encountered when streaming media over those existing networks.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9317.

Copyright Notice

Table of Contents

1. Introduction
   1.1. Key Definitions
   1.2. Document Scope
2. Our Focus on Streaming Video
3. Bandwidth Provisioning
   3.1. Scaling Requirements for Media Delivery
      3.1.1. Video Bitrates
      3.1.2. Virtual Reality Bitrates
   3.2. Path Bottlenecks and Constraints
      3.2.1. Recognizing Changes from a Baseline
   3.3. Path Requirements
   3.4. Caching Systems
   3.5. Predictable Usage Profiles
   3.6. Unpredictable Usage Profiles
      3.6.1. Peer-to-Peer Applications
      3.6.2. Impact of Global Pandemic
4. Latency Considerations
   4.1. Ultra-Low-Latency
      4.1.1. Near-Real-Time Latency
   4.2. Low-Latency Live
   4.3. Non-Low-Latency Live
   4.4. On-Demand

5. Adaptive Encoding, Adaptive Delivery, and Measurement Collection
5.1. Overview
5.2. Adaptive Encoding
5.3. Adaptive Segmented Delivery
5.4. Advertising
5.5. Bitrate Detection Challenges
5.5.1. Idle Time between Segments
5.5.2. Noisy Measurements
5.5.3. Wide and Rapid Variation in Path Capacity
5.6. Measurement Collection
6. Transport Protocol Behaviors and Their Implications for Media Transport Protocols
6.1. Media Transport over Reliable Transport Protocols
6.2. Media Transport over Unreliable Transport Protocols
6.3. QUIC and Changing Transport Protocol Behavior
7. Streaming Encrypted Media
7.1. General Considerations for Streaming Media Encryption
7.2. Considerations for Hop-by-Hop Media Encryption
7.3. Considerations for End-to-End Media Encryption
8. Additional Resources for Streaming Media
9. IANA Considerations
10. Security Considerations
11. Informative References
Acknowledgments
Authors' Addresses

1. Introduction

This document provides an overview of operational networking and transport protocol issues that pertain to the quality of experience (QoE) when streaming video and other high-bitrate media over the Internet.

This document is intended to explain the characteristics of streaming media delivery that have surprised network designers or transport experts who lack specific media expertise, since streaming media highlights key differences between common assumptions in existing networking practices and observations of media delivery issues encountered when streaming media over those existing networks.

1.1. Key Definitions

This document defines "high-bitrate streaming media over the Internet" as follows:

- "High-bitrate" is a context-sensitive term broadly intended to capture rates that can be sustained over some but not all of the target audience's network connections. A snapshot of values commonly qualifying as high-bitrate on today's Internet is given by the higher-value entries in Section 3.1.1.
- "Streaming" means the continuous transmission of media segments from a server to a client and its simultaneous consumption by the client.
  - The term "simultaneous" is critical, as media segment transmission is not considered "streaming" if one downloads a media file and plays it after the download is completed. Instead, this would be called "download and play".
  - This has two implications. First, the sending rate for media segments must match the client's consumption rate (whether loosely or tightly) to provide uninterrupted playback. That is, the client must not run out of media segments (buffer underrun) and must not accept more media segments than it can buffer before playback (buffer overrun).
  - Second, the client's media segment consumption rate is limited not only by the path's available bandwidth but also by media segment availability. The client cannot fetch media segments that a media server cannot provide (yet).
- "Media" refers to any type of media and associated streams, such as video, audio, metadata, etc.
- "Over the Internet" means that a single operator does not have control of the entire path between media servers and media clients, so it is not a "walled garden".

This document uses these terms to describe the streaming media ecosystem:

Streaming Media Operator: an entity that provides streaming media servers

Media Server: a server that provides streaming media to a media player, which is also referred to as a streaming media server, or simply a server

Intermediary: an entity that is on-path, between the streaming media operator and the ultimate media consumer, and that is media aware

When the streaming media is encrypted, an intermediary must have credentials that allow the intermediary to decrypt the media in order to be media aware.

An intermediary can be one of many specialized subtypes that meet this definition.

Media Player: an endpoint that requests streaming media from a media server for an ultimate media consumer, which is also referred to as a streaming media client, or simply a client

Ultimate Media Consumer: a human or machine using a media player

1.2. Document Scope

A full review of all streaming media considerations for all types of media over all types of network paths is too broad a topic to cover comprehensively in a single document.

This document focuses chiefly on the large-scale delivery of streaming high-bitrate media to end users. It is primarily intended for those controlling endpoints involved in delivering streaming media traffic. This can include origin servers publishing content, intermediaries like content delivery networks (CDNs), and providers for client devices and media players.

Most of the considerations covered in this document apply to both "live media" (created and streamed as an event is in progress) and "media on demand" (previously recorded media that is streamed from storage), except where noted.

Most of the considerations covered in this document apply to both media that is consumed by a media player, for viewing by a human, and media that is consumed by a machine, such as a media recorder that is executing an adaptive bitrate (ABR) streaming algorithm, except where noted.

This document contains

- a short description of streaming video characteristics in Section 2 to set the stage for the rest of the document,
- general guidance on bandwidth provisioning (Section 3) and latency considerations (Section 4) for streaming media delivery,
- a description of adaptive encoding and adaptive delivery techniques in common use for streaming video, along with a description of the challenges media senders face in detecting the bitrate available between the media sender and media receiver, and a collection of measurements by a third party for use in analytics (Section 5),
- a description of existing transport protocols used for media streaming and the issues encountered when using those protocols, along with a description of the QUIC transport protocol [RFC9000] more recently used for streaming media (Section 6),
- a description of implications when streaming encrypted media (Section 7), and
- a pointer to additional resources for further reading on this rapidly changing subject (Section 8).

Topics outside this scope include the following:

- an in-depth examination of real-time, two-way interactive media, such as videoconferencing; although this document touches lightly on topics related to this space, the intent is to let readers know that for more in-depth coverage they should look to other documents, since

the techniques and issues for interactive real-time, two-way media differ so dramatically from those in large-scale, one-way delivery of streaming media.

- specific recommendations on operational practices to mitigate issues described in this document; although some known mitigations are mentioned in passing, the primary intent is to provide a point of reference for future solution proposals to describe how new technologies address or avoid existing problems.
- generalized network performance techniques; while considerations, such as data center design, transit network design, and "walled garden" optimizations, can be crucial components of a performant streaming media service, these are considered independent topics that are better addressed by other documents.
- transparent tunnels; while tunnels can have an impact on streaming media via issues like the round-trip time and the maximum transmission unit (MTU) of packets carried over tunnels, for the purposes of this document, these issues are considered as part of the set of network path properties.

Questions about whether this document also covers "Web Real-Time Communication (WebRTC)" have come up often. It does not. WebRTC's principal media transport protocol [RFC8834] [RFC8835], the Real-time Transport Protocol (RTP), is mentioned in this document. However, as noted in Section 2, it is difficult to give general guidance for unreliable media transport protocols used to carry interactive real-time media.

2. Our Focus on Streaming Video

As the Internet has grown, an increasingly large share of the traffic delivered to end users has become video. The most recent available estimates found that 75% of the total traffic to end users was video in 2019 (as described in [RFC8404], such traffic surveys have since become impossible to conduct due to ubiquitous encryption). At that time, the share of video traffic had been growing for years and was projected to continue growing (Appendix D of [CVNI]).

A substantial part of this growth is due to the increased use of streaming video. However, video traffic in real-time communications (for example, online videoconferencing) has also grown significantly. While both streaming video and videoconferencing have real-time delivery and latency requirements, these requirements vary from one application to another. For additional discussion of latency requirements, see Section 4.

In many contexts, media traffic can be handled transparently as generic application-level traffic. However, as the volume of media traffic continues to grow, it is becoming increasingly important to consider the effects of network design decisions on application-level performance, with considerations for the impact on media delivery.

Much of the focus of this document is on media streaming over HTTP. HTTP is widely used for media streaming because

- support for HTTP is widely available in a wide range of operating systems,
- HTTP is also used in a wide variety of other applications,
- HTTP has been demonstrated to provide acceptable performance over the open Internet,

- HTTP includes state-of-the-art standardized security mechanisms, and
- HTTP can use already-deployed caching infrastructure, such as CDNs, local proxies, and browser caches.

Various HTTP versions have been used for media delivery. HTTP/1.0, HTTP/1.1, and HTTP/2 are carried over TCP [RFC9293], and TCP's transport behavior is described in Section 6.1. HTTP/3 is carried over QUIC, and QUIC's transport behavior is described in Section 6.3.

Unreliable media delivery using RTP and other UDP-based protocols is also discussed in Sections 4.1, 6.2, and 7.2, but it is difficult to give general guidance for these applications. For instance, when packet loss occurs, the most appropriate response may depend on the type of codec being used.

3. Bandwidth Provisioning

3.1. Scaling Requirements for Media Delivery

3.1.1. Video Bitrates

Video bitrate selection depends on many variables including the resolution (height and width), frame rate, color depth, codec, encoding parameters, scene complexity, and amount of motion. Generally speaking, as the resolution, frame rate, color depth, scene complexity, and amount of motion increase, the encoding bitrate increases. As newer codecs with better compression tools are used, the encoding bitrate decreases. Similarly, a multi-pass encoding generally produces better quality output compared to single-pass encoding at the same bitrate or delivers the same quality at a lower bitrate.

Here are a few common resolutions used for video content, with typical ranges of bitrates for the two most popular video codecs [Encodings].

Name	Width x Height	H.264	H.265
DVD	720 x 480	1.0 Mbps	0.5 Mbps
720p (1K)	1280 x 720	3-4.5 Mbps	2-4 Mbps
1080p (2K)	1920 x 1080	6-8 Mbps	4.5-7 Mbps
2160p (4k)	3840 x 2160	N/A	10-20 Mbps

Table 1: Typical Resolutions and Bitrate Ranges Used for Video Encoding

- Note that these codecs do not take the actual "available bandwidth" between media servers and media players into account when encoding because the codec does not have any idea what network paths and network path conditions will carry the encoded video at some point in the future. It is common for codecs to offer a small number of resource variants, differing only in the bandwidth each variant targets.

- Note that media players attempting to receive encoded video across a network path with insufficient available path bandwidth might request the media server to provide video encoded for lower bitrates, at the cost of lower video quality, as described in Section 5.3.
- In order to provide multiple encodings for video resources, the codec must produce multiple variants (also called renditions) of the video resource encoded at various bitrates, as described in Section 5.2.

3.1.2. Virtual Reality Bitrates

The bitrates given in Section 3.1.1 describe video streams that provide the user with a single, fixed point of view -- therefore, the user has no "degrees of freedom", and the user sees all of the video image that is available.

Even basic virtual reality (360-degree) videos that allow users to look around freely (referred to as "three degrees of freedom" or 3DoF) require substantially larger bitrates when they are captured and encoded, as such videos require multiple fields of view of the scene. Yet, due to smart delivery methods, such as viewport-based or tile-based streaming, there is no need to send the whole scene to the user. Instead, the user needs only the portion corresponding to its viewpoint at any given time [Survey360].

In more immersive applications, where limited user movement ("three degrees of freedom plus" or 3DoF+) or full user movement ("six degrees of freedom" or 6DoF) is allowed, the required bitrate grows even further. In this case, immersive content is typically referred to as volumetric media. One way to represent the volumetric media is to use point clouds, where streaming a single object may easily require a bitrate of 30 Mbps or higher. Refer to [MPEGI] and [PCC] for more details.

3.2. Path Bottlenecks and Constraints

Even when the bandwidth requirements for media streams along a path are well understood, additional analysis is required to understand the constraints on bandwidth at various points along the path between media servers and media players. Media streams can encounter bottlenecks at many points along a path, whether the bottleneck happens at a node or at a path segment along the path, and these bottlenecks may involve a lack of processing power, buffering capacity, link speed, or any other exhaustible resource.

Media servers may react to bandwidth constraints using two independent feedback loops:

- Media servers often respond to application-level feedback from the media player that indicates a bottleneck somewhere along the path by sending a different media bitrate. This is described in greater detail in Section 5.
- Media servers also typically rely on transport protocols with capacity-seeking congestion controllers that probe for available path bandwidth and adjust the media sending rate based on transport mechanisms. This is described in greater detail in Section 6.

The result is that these two (potentially competing) "helpful" mechanisms each respond to the same bottleneck with no coordination between themselves, so that each is unaware of actions taken by the other, and this can result in QoE for users that is significantly lower than what could have been achieved.

One might wonder why media servers and transport protocols are each unaware of what the other is doing, and there are multiple reasons for that. One reason is that media servers are often implemented as applications executing in user space, relying on a general-purpose operating system that typically has its transport protocols implemented in the operating system kernel, making decisions that the media server never knows about.

As one example, if a media server overestimates the available bandwidth to the media player,

- the transport protocol may detect loss due to congestion and reduce its sending window size per round trip,
- the media server adapts to application-level feedback from the media player and reduces its own sending rate, and/or
- the transport protocol sends media at the new, lower rate and confirms that this new, lower rate is "safe" because no transport-level loss is occurring.

However, because the media server continues to send at the new, lower rate, the transport protocol's maximum sending rate is now limited by the amount of information the media server queues for transmission. Therefore, the transport protocol cannot probe for available path bandwidth by sending at a higher rate until the media player requests segments that buffer enough data for the transport to perform the probing.

To avoid these types of situations, which can potentially affect all the users whose streaming media segments traverse a bottleneck path segment, there are several possible mitigations that streaming operators can use. However, the first step toward mitigating a problem is knowing that a problem is occurring.

3.2.1. Recognizing Changes from a Baseline

There are many reasons why path characteristics might change in normal operation. For example:

- If the path topology changes. For example, routing changes, which can happen in normal operation, may result in traffic being carried over a new path topology that is partially or entirely disjointed from the previous path, especially if the new path topology includes one or more path segments that are more heavily loaded, offer lower total bandwidth, change the overall Path MTU size, or simply cover more distance between the path endpoints.
- If cross traffic that also traverses part or all of the same path topology increases or decreases, especially if this new cross traffic is "inelastic" and does not respond to indications of path congestion.
- Wireless links (Wi-Fi, 5G, LTE, etc.) may see rapid changes to capacity from changes in radio interference and signal strength as endpoints move.

To recognize that a path carrying streaming media has experienced a change, maintaining a baseline that captures its prior properties is fundamental. Analytics that aid in that recognition can be more or less sophisticated and can usefully operate on several different time scales, from milliseconds to hours or days.

Useful properties to monitor for changes can include the following:

- round-trip times
- loss rate (and explicit congestion notification (ECN) [RFC3168] when in use)
- out-of-order packet rate
- packet and byte receive rate
- application-level goodput
- properties of other connections carrying competing traffic, in addition to the connections carrying the streaming media
- externally provided measurements, for example, from network cards or metrics collected by the operating system

3.3. Path Requirements

The bitrate requirements in Section 3.1 are per end user actively consuming a media feed, so in the worst case, the bitrate demands can be multiplied by the number of simultaneous users to find the bandwidth requirements for a delivery path with that number of users downstream. For example, at a node with 10,000 downstream users simultaneously consuming video streams, approximately 80 Gbps might be necessary for all of them to get typical content at 1080p resolution.

However, when there is some overlap in the feeds being consumed by end users, it is sometimes possible to reduce the bandwidth provisioning requirements for the network by performing some kind of replication within the network. This can be achieved via object caching with the delivery of replicated objects over individual connections and/or by packet-level replication using multicast.

To the extent that replication of popular content can be performed, bandwidth requirements at peering or ingest points can be reduced to as low as a per-feed requirement instead of a per-user requirement.

3.4. Caching Systems

When demand for content is relatively predictable, and especially when that content is relatively static, caching content close to requesters and preloading caches to respond quickly to initial requests are often useful (for example, HTTP/1.1 caching is described in [RFC9111]). This is subject to the usual considerations for caching -- for example, how much data must be cached to make a significant difference to the requester and how the benefit of caching and preloading cache balances against the costs of tracking stale content in caches and refreshing that content.

It is worth noting that not all high-demand content is "live" content. One relevant example is when popular streaming content can be staged close to a significant number of requesters, as can happen when a new episode of a popular show is released. This content may be largely stable and is therefore low-cost to maintain in multiple places throughout the Internet. This can reduce demands for high end-to-end bandwidth without having to use mechanisms like multicast.

Caching and preloading can also reduce exposure to peering point congestion, since less traffic crosses the peering point exchanges if the caches are placed in peer networks. This is especially true when the content can be preloaded during off-peak hours and if the transfer can make use of "A Lower-Effort Per-Hop Behavior (LE PHB) for Differentiated Services" [RFC8622], "Low Extra Delay Background Transport (LEDBAT)" [RFC6817], or similar mechanisms.

All of this depends, of course, on the ability of a streaming media operator to predict usage and provision bandwidth, caching, and other mechanisms to meet the needs of users. In some cases (Section 3.5), this is relatively routine, but in other cases, it is more difficult (Section 3.6).

With the emergence of ultra-low-latency streaming, responses have to start streaming to the end user while still being transmitted to the cache and while the cache does not yet know the size of the object. Some of the popular caching systems were designed around a cache footprint and had deeply ingrained assumptions about knowing the size of objects that are being stored, so the change in design requirements in long-established systems caused some errors in production. Incidents occurred where a transmission error in the connection from the upstream source to the cache could result in the cache holding a truncated segment and transmitting it to the end user's device. In this case, players rendering the stream often had a playback freeze until the player was reset. In some cases, the truncated object was even cached that way and served later to other players as well, causing continued stalls at the same spot in the media for all players playing the segment delivered from that cache node.

3.5. Predictable Usage Profiles

Historical data shows that users consume more videos, and these videos are encoded at a bitrate higher than they were in the past. Improvements in the codecs that help reduce the encoding bitrates with better compression algorithms have not offset the increase in the demand for the higher quality video (higher resolution, higher frame rate, better color gamut, better dynamic range, etc.). In particular, mobile data usage in cellular access networks has shown a large jump over the years due to increased consumption of entertainment and conversational video.

3.6. Unpredictable Usage Profiles

It is also possible for usage profiles to change significantly and suddenly. These changes are more difficult to plan for, but at a minimum, recognizing that sudden changes are happening is critical.

The two examples that follow are instructive.

3.6.1. Peer-to-Peer Applications

In the first example, described in "Report from the IETF Workshop on Peer-to-Peer (P2P) Infrastructure, May 28, 2008" [RFC5594], when the BitTorrent file sharing application came into widespread use in 2005, sudden and unexpected growth in peer-to-peer traffic led to complaints from ISP customers about the performance of delay-sensitive traffic (Voice over IP (VoIP) and gaming). These performance issues resulted from at least two causes:

- Many access networks for end users used underlying technologies that are inherently asymmetric, favoring downstream bandwidth (e.g., ADSL, cellular technologies, and most IEEE 802.11 variants), assuming that most users will need more downstream bandwidth than upstream bandwidth. This is a good assumption for client-server applications, such as streaming media or software downloads, but BitTorrent rewarded peers that uploaded as much as they downloaded, so BitTorrent users had much more symmetric usage profiles, which interacted badly with these asymmetric access network technologies.
- Some P2P systems also used distributed hash tables to organize peers into a ring topology, where each peer knew its "next peer" and "previous peer". There was no connection between the application-level ring topology and the lower-level network topology, so a peer's "next peer" might be anywhere on the reachable Internet. Traffic models that expected most communication to take place with a relatively small number of servers were unable to cope with peer-to-peer traffic that was much less predictable.

Especially as end users increase the use of video-based social networking applications, it will be helpful for access network providers to watch for increasing numbers of end users uploading significant amounts of content.

3.6.2. Impact of Global Pandemic

Early in 2020, the COVID-19 pandemic and resulting quarantines and shutdowns led to significant changes in traffic patterns due to a large number of people who suddenly started working and attending school remotely and using more interactive applications (e.g., videoconferencing and streaming media). Subsequently, the Internet Architecture Board (IAB) held a COVID-19 Network Impacts Workshop [RFC9075] in November 2020. The following observations from the workshop report are worth considering.

- Participants describing different types of networks reported different kinds of impacts, but all types of networks saw impacts.
- Mobile networks saw traffic reductions, and residential networks saw significant increases.
- Reported traffic increases from ISPs and Internet Exchange Points (IXPs) over just a few weeks were as big as the traffic growth over the course of a typical year, representing a 15-20% surge in growth to land at a new normal that was much higher than anticipated.
- At Deutscher Commercial Internet Exchange (DE-CIX) Frankfurt, the world's largest IXP in terms of data throughput, the year 2020 has seen the largest increase in peak traffic within a single year since the IXP was founded in 1995.
- The usage pattern changed significantly as work-from-home and videoconferencing usage peaked during normal work hours, which would have typically been off-peak hours with

adults at work and children at school. One might expect that the peak would have had more impact on networks if it had happened during typical evening peak hours for streaming applications.

- The increase in daytime bandwidth consumption reflected both significant increases in essential applications, such as videoconferencing and virtual private networks (VPNs), and entertainment applications as people watched videos or played games.
- At the IXP level, it was observed that physical link utilization increased. This phenomenon could probably be explained by a higher level of uncacheable traffic, such as videoconferencing and VPNs, from residential users as they stopped commuting and switched to working at home.

Again, it will be helpful for streaming operators to monitor traffic as described in Section 5.6, watching for sudden changes in performance.

4. Latency Considerations

Streaming media latency refers to the "glass-to-glass" time duration, which is the delay between the real-life occurrence of an event and the streamed media being appropriately played on an end user's device. Note that this is different from the network latency (defined as the time for a packet to cross a network from one end to another end) because it includes media encoding/decoding and buffering time and, for most cases, also the ingest to an intermediate service, such as a CDN or other media distribution service, rather than a direct connection to an end user.

The team working on this document found these rough categories to be useful when considering a streaming media application's latency requirements:

- ultra-low-latency (less than 1 second)
- low-latency live (less than 10 seconds)
- non-low-latency live (10 seconds to a few minutes)
- on-demand (hours or more)

4.1. Ultra-Low-Latency

Ultra-low-latency delivery of media is defined here as having a glass-to-glass delay target under 1 second.

Some media content providers aim to achieve this level of latency for live media events. This introduces new challenges when compared to the other latency categories described in Section 4, because ultra-low-latency is on the same scale as commonly observed end-to-end network latency variation, often due to bufferbloat [CoDel], Wi-Fi error correction, or packet reordering. These effects can make it difficult to achieve ultra-low-latency for many users and may require accepting relatively frequent user-visible media artifacts. However, for controlled environments that provide mitigations against such effects, ultra-low-latency is potentially achievable with the right provisioning and the right media transport technologies.

Most applications operating over IP networks and requiring latency this low use the Real-time Transport Protocol (RTP) [RFC3550] or WebRTC [RFC8825], which uses RTP as its media transport protocol, along with several other protocols necessary for safe operation in browsers.

It is worth noting that many applications for ultra-low-latency delivery do not need to scale to as many users as applications for low-latency and non-low-latency live delivery, which simplifies many delivery considerations.

Recommended reading for applications adopting an RTP-based approach also includes [RFC7656]. For increasing the robustness of the playback by implementing adaptive playout methods, refer to [RFC4733] and [RFC6843].

4.1.1. Near-Real-Time Latency

Some Internet applications that incorporate media streaming have specific interactivity or control-feedback requirements that drive much lower glass-to-glass media latency targets than 1 second. These include videoconferencing or voice calls; remote video gameplay; remote control of hardware platforms like drones, vehicles, or surgical robots; and many other envisioned or deployed interactive applications.

Applications with latency targets in these regimes are out of scope for this document.

4.2. Low-Latency Live

Low-latency live delivery of media is defined here as having a glass-to-glass delay target under 10 seconds.

This level of latency is targeted to have a user experience similar to broadcast TV delivery. A frequently cited problem with failing to achieve this level of latency for live sporting events is the user experience failure from having crowds within earshot of one another who react audibly to an important play or from users who learn of an event in the match via some other channel, for example, social media, before it has happened on the screen showing the sporting event.

Applications requiring low-latency live media delivery are generally feasible at scale with some restrictions. This typically requires the use of a premium service dedicated to the delivery of live media, and some trade-offs may be necessary relative to what is feasible in a higher-latency service. The trade-offs may include higher costs, delivering a lower quality media, reduced flexibility for adaptive bitrates, or reduced flexibility for available resolutions so that fewer devices can receive an encoding tuned for their display. Low-latency live delivery is also more susceptible to user-visible disruptions due to transient network conditions than higher-latency services.

Implementation of a low-latency live media service can be achieved with the use of HTTP Live Streaming (HLS) [RFC8216] by using its low-latency extension (called LL-HLS) [HLS-RFC8216BIS] or with Dynamic Adaptive Streaming over HTTP (DASH) [MPEG-DASH] by using its low-latency extension (called LL-DASH) [LL-DASH]. These extensions use the Common Media Application Format (CMAF) standard [MPEG-CMAF] that allows the media to be packaged into and transmitted in units smaller than segments, which are called "chunks" in CMAF language. This

way, the latency can be decoupled from the duration of the media segments. Without a CMAF-like packaging, lower latencies can only be achieved by using very short segment durations. However, using shorter segments means using more frequent intra-coded frames, and that is detrimental to video encoding quality. The CMAF standard allows us to still use longer segments (improving encoding quality) without penalizing latency.

While an LL-HLS client retrieves each chunk with a separate HTTP GET request, an LL-DASH client uses the chunked transfer encoding feature of the HTTP [CMAF-CTE], which allows the LL-DASH client to fetch all the chunks belonging to a segment with a single GET request. An HTTP server can transmit the CMAF chunks to the LL-DASH client as they arrive from the encoder/packager. A detailed comparison of LL-HLS and LL-DASH is given in [MMSP20].

4.3. Non-Low-Latency Live

Non-low-latency live delivery of media is defined here as a live stream that does not have a latency target shorter than 10 seconds.

This level of latency is the historically common case for segmented media delivery using HLS and DASH. This level of latency is often considered adequate for content like news. This level of latency is also sometimes achieved as a fallback state when some part of the delivery system or the client-side players do not support low-latency live streaming.

This level of latency can typically be achieved at scale with commodity CDN services for HTTP(s) delivery, and in some cases, the increased time window can allow for the production of a wider range of encoding options relative to the requirements for a lower-latency service without the need for increasing the hardware footprint, which can allow for wider device interoperability.

4.4. On-Demand

On-demand media streaming refers to the playback of pre-recorded media based on a user's action. In some cases, on-demand media is produced as a by-product of a live media production, using the same segments as the live event but freezing the manifest that describes the media available from the media server after the live event has finished. In other cases, on-demand media is constructed out of pre-recorded assets with no streaming necessarily involved during the production of the on-demand content.

On-demand media generally is not subject to latency concerns, but other timing-related considerations can still be as important or even more important to the user experience than the same considerations with live events. These considerations include the startup time, the stability of the media stream's playback quality, and avoidance of stalls and other media artifacts during the playback under all but the most severe network conditions.

In some applications, optimizations are available to on-demand media but are not always available to live events, such as preloading the first segment for a startup time that does not have to wait for a network download to begin.

5. Adaptive Encoding, Adaptive Delivery, and Measurement Collection

This section describes one of the best-known ways to provide a good user experience over a given network path, but one thing to keep in mind is that application-level mechanisms cannot provide a better experience than the underlying network path can support.

5.1. Overview

A simple model of media playback can be described as a media stream consumer, a buffer, and a transport mechanism that fills the buffer. The consumption rate is fairly static and is represented by the content bitrate. The size of the buffer is also commonly a fixed size. The buffer fill process needs to be at least fast enough to ensure that the buffer is never empty; however, it also can have significant complexity when things like personalization or advertising insertion workflows are introduced.

The challenges in filling the buffer in a timely way fall into two broad categories:

- Content variation (also sometimes called a "bitrate ladder") is the set of content renditions that are available at any given selection point.
- Content selection comprises all of the steps a client uses to determine which content rendition to play.

The mechanism used to select the bitrate is part of the content selection, and the content variation is all of the different bitrate renditions.

Adaptive bitrate streaming ("ABR streaming" or simply "ABR") is a commonly used technique for dynamically adjusting the media quality of a stream to match bandwidth availability. When this goal is achieved, the media server will tend to send enough media that the media player does not "stall", without sending so much media that the media player cannot accept it.

ABR uses an application-level response strategy in which the streaming client attempts to detect the available bandwidth of the network path by first observing the successful application-layer download speed; then, given the available bandwidth, the client chooses a bitrate for each of the video, audio, subtitles, and metadata (among a limited number of available options for each type of media) that fits within that bandwidth, typically adjusting as changes in available bandwidth occur in the network or changes in capabilities occur during the playback (such as available memory, CPU, display size, etc.).

5.2. Adaptive Encoding

Media servers can provide media streams at various bitrates because the media has been encoded at various bitrates. This is a so-called "ladder" of bitrates that can be offered to media players as part of the manifest so that the media player can select among the available bitrate choices.

The media server may also choose to alter which bitrates are made available to players by adding or removing bitrate options from the ladder delivered to the player in subsequent manifests built and sent to the player. This way, both the player, through its selection of bitrate to request from the manifest, and the server, through its construction of the bitrates offered in the manifest, are able to affect network utilization.

5.3. Adaptive Segmented Delivery

Adaptive segmented delivery attempts to optimize its own use of the path between a media server and a media client. ABR playback is commonly implemented by streaming clients using HLS [RFC8216] or DASH [MPEG-DASH] to perform a reliable segmented delivery of media over HTTP. Different implementations use different strategies [ABRSurvey], often relying on proprietary algorithms (called rate adaptation or bitrate selection algorithms) to perform available bandwidth estimation/prediction and the bitrate selection.

Many systems will do an initial probe or a very simple throughput speed test at the start of media playback. This is done to get a rough sense of the highest (total) media bitrate that the network between the server and player will likely be able to provide under initial network conditions. After the initial testing, clients tend to rely upon passive network observations and will make use of player-side statistics, such as buffer fill rates, to monitor and respond to changing network conditions.

The choice of bitrate occurs within the context of optimizing for one or more metrics monitored by the client, such as the highest achievable audiovisual quality or the lowest chances for a rebuffering event (playback stall).

5.4. Advertising

The inclusion of advertising alongside or interspersed with streaming media content is common in today's media landscape.

Some commonly used forms of advertising can introduce potential user experience issues for a media stream. This section provides a very brief overview of a complex and rapidly evolving space.

The same techniques used to allow a media player to switch between renditions of different bitrates at segment boundaries can also be used to enable the dynamic insertion of advertisements (hereafter referred to as "ads"), but this does not mean that the insertion of ads has no effect on the user's quality of experience.

Ads may be inserted with either Client-side Ad Insertion (CSAI) or Server-side Ad Insertion (SSAI). In CSAI, the ABR manifest will generally include links to an external ad server for some segments of the media stream, while in SSAI, the server will remain the same during ads but will include media segments that contain the advertising. In SSAI, the media segments may or may not be sourced from an external ad server like with CSAI.

In general, the more targeted the ad request is, the more requests the ad service needs to be able to handle concurrently. If connectivity is poor to the ad service, this can cause rebuffering even if the underlying media assets (both content and ads) can be accessed quickly. The less targeted the ad request is, the more likely that ad requests can be consolidated and that ads can be cached similarly to the media content.

In some cases, especially with SSAI, advertising space in a stream is reserved for a specific advertiser and can be integrated with the video so that the segments share the same encoding properties, such as bitrate, dynamic range, and resolution. However, in many cases, ad servers integrate with a Supply Side Platform (SSP) that offers advertising space in real-time auctions via an Ad Exchange, with bids for the advertising space coming from Demand Side Platforms (DSPs) that collect money from advertisers for delivering the ads. Most such Ad Exchanges use application-level protocol specifications published by the Interactive Advertising Bureau [IAB-ADS], an industry trade organization.

This ecosystem balances several competing objectives, and integrating with it naively can produce surprising user experience results. For example, ad server provisioning and/or the bitrate of the ad segments might be different from that of the main content, and either of these differences can result in playback stalls. For another example, since the inserted ads are often produced independently, they might have a different base volume level than the main content, which can make for a jarring user experience.

Another major source of competing objectives comes from user privacy considerations vs. the advertiser's incentives to target ads to user segments based on behavioral data. Multiple studies, for example, [BEHAVE] and [BEHAVE2], have reported large improvements in ad effectiveness when using behaviorally targeted ads, relative to untargeted ads. This provides a strong incentive for advertisers to gain access to the data necessary to perform behavioral targeting, leading some to engage in what is indistinguishable from a pervasive monitoring attack [RFC7258] based on user tracking in order to collect the relevant data. A more complete review of issues in this space is available in [BALANCING].

On top of these competing objectives, this market historically has had incidents of misreporting of ad delivery to end users for financial gain [ADFRAUD]. As a mitigation for concerns driven by those incidents, some SSPs have required the use of specific media players that include features like reporting of ad delivery or providing additional user information that can be used for tracking.

In general, this is a rapidly developing space with many considerations, and media streaming operators engaged in advertising may need to research these and other concerns to find solutions that meet their user experience, user privacy, and financial goals. For further reading on mitigations, [BAP] has published some standards and best practices based on user experience research.

5.5. Bitrate Detection Challenges

This kind of bandwidth-measurement system can experience various troubles that are affected by networking and transport protocol issues. Because adaptive application-level response strategies are often using rates as observed by the application layer, there are sometimes inscrutable transport-level protocol behaviors that can produce surprising measurement values when the application-level feedback loop is interacting with a transport-level feedback loop.

A few specific examples of surprising phenomena that affect bitrate detection measurements are described in the following subsections. As these examples will demonstrate, it is common to encounter cases that can deliver application-level measurements that are too low, too high, and (possibly) correct but that vary more quickly than a lab-tested selection algorithm might expect.

These effects and others that cause transport behavior to diverge from lab modeling can sometimes have a significant impact on bitrate selection and on user QoE, especially where players use naive measurement strategies and selection algorithms that do not account for the likelihood of bandwidth measurements that diverge from the true path capacity.

5.5.1. Idle Time between Segments

When the bitrate selection is chosen substantially below the available capacity of the network path, the response to a segment request will typically complete in much less absolute time than the duration of the requested segment, leaving significant idle time between segment downloads. This can have a few surprising consequences:

- TCP slow-start, when restarting after idle, requires multiple RTTs to re-establish a throughput at the network's available capacity. When the active transmission time for segments is substantially shorter than the time between segments, leaving an idle gap between segments that triggers a restart of TCP slow-start, the estimate of the successful download speed coming from the application-visible receive rate on the socket can thus end up much lower than the actual available network capacity. This, in turn, can prevent a shift to the most appropriate bitrate. [RFC7661] provides some mitigations for this effect at the TCP transport layer for senders who anticipate a high incidence of this problem.
- Mobile flow-bandwidth spectrum and timing mapping can be impacted by idle time in some networks. The carrier capacity assigned to a physical or virtual link can vary with activity. Depending on the idle time characteristics, this can result in a lower available bitrate than would be achievable with a steadier transmission in the same network.

Some receiver-side ABR algorithms, such as [ELASTIC], are designed to try to avoid this effect.

Another way to mitigate this effect is by the help of two simultaneous TCP connections, as explained in [MMSys11] for Microsoft Smooth Streaming. In some cases, the system-level TCP slow-start restart can also be disabled, for example, as described in [OReilly-HPBN].

5.5.2. Noisy Measurements

In addition to smoothing over an appropriate time scale to handle network jitter (see [RFC5481]), ABR systems relying on measurements at the application layer also have to account for noise from the in-order data transmission at the transport layer.

For instance, in the event of a lost packet on a TCP connection with SACK support (a common case for segmented delivery in practice), loss of a packet can provide a confusing bandwidth signal to the receiving application. Because of the sliding window in TCP, many packets may be accepted by the receiver without being available to the application until the missing packet arrives. Upon the arrival of the one missing packet after retransmit, the receiver will suddenly get access to a lot of data at the same time.

To a receiver measuring bytes received per unit time at the application layer and interpreting it as an estimate of the available network bandwidth, this appears as a high jitter in the goodput measurement, presenting as a stall followed by a sudden leap that can far exceed the actual capacity of the transport path from the server when the hole in the received data is filled by a later retransmission.

5.5.3. Wide and Rapid Variation in Path Capacity

As many end devices have moved to wireless connections for the final hop (such as Wi-Fi, 5G, LTE, etc.), new problems in bandwidth detection have emerged.

In most real-world operating environments, wireless links can often experience sudden changes in capacity as the end user device moves from place to place or encounters new sources of interference. Microwave ovens, for example, can cause a throughput degradation in Wi-Fi of more than a factor of 2 while active [Micro].

These swings in actual transport capacity can result in user experience issues when interacting with ABR algorithms that are not tuned to handle the capacity variation gracefully.

5.6. Measurement Collection

Media players use measurements to guide their segment-by-segment adaptive streaming requests but may also provide measurements to streaming media providers.

In turn, media providers may base analytics on these measurements to guide decisions, such as whether adaptive encoding bitrates in use are the best ones to provide to media players or whether current media content caching is providing the best experience for viewers.

To that effect, the Consumer Technology Association (CTA), who owns the Web Application Video Ecosystem (WAVE) project, has published two important specifications.

- CTA-2066: Streaming Quality of Experience Events, Properties and Metrics

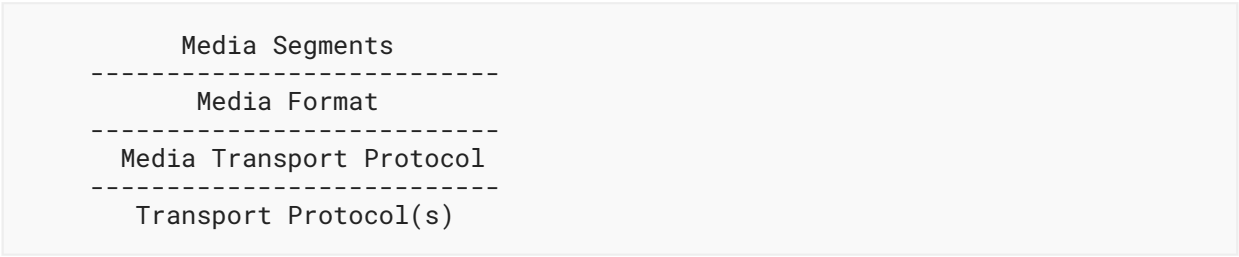
[CTA-2066] specifies a set of media player events, properties, QoE metrics, and associated terminology for representing streaming media QoE across systems, media players, and analytics vendors. While all these events, properties, metrics, and associated terminology are used across a number of proprietary analytics and measurement solutions, they were used in slightly (or vastly) different ways that led to interoperability issues. CTA-2066 attempts to address this issue by defining common terminology and how each metric should be computed for consistent reporting.

- CTA-5004: Web Application Video Ecosystem - Common Media Client Data (CMCD)

Many assume that the CDNs have a holistic view of the health and performance of the streaming clients. However, this is not the case. The CDNs produce millions of log lines per second across hundreds of thousands of clients, and they have no concept of a "session" as a client would have, so CDNs are decoupled from the metrics the clients generate and report. A CDN cannot tell which request belongs to which playback session, the duration of any media object, the bitrate, or whether any of the clients have stalled and are rebuffering or are about to stall and will rebuffer. The consequence of this decoupling is that a CDN cannot prioritize delivery for when the client needs it most, prefetch content, or trigger alerts when the network itself may be underperforming. One approach to couple the CDN to the playback sessions is for the clients to communicate standardized media-relevant information to the CDNs while they are fetching data. [CTA-5004] was developed exactly for this purpose.

6. Transport Protocol Behaviors and Their Implications for Media Transport Protocols

Within this document, the term "media transport protocol" is used to describe any protocol that carries media metadata and media segments in its payload, and the term "transport protocol" describes any protocol that carries a media transport protocol, or another transport protocol, in its payload. This is easier to understand if the reader assumes a protocol stack that looks something like this:



where

- "Media segments" would be something like the output of a codec or some other source of media segments, such as closed-captioning,
- "Media format" would be something like an RTP payload format [RFC2736] or an ISO base media file format (ISOBMFF) profile [ISOBMFF],

- "Media transport protocol" would be something like RTP [RFC3550] or DASH [MPEG-DASH], and
- "Transport protocol" would be a protocol that provides appropriate transport services, as described in Section 5 of [RFC8095].

Not all possible streaming media applications follow this model, but for the ones that do, it seems useful to distinguish between the protocol layer that is aware it is transporting media segments and underlying protocol layers that are not aware.

As described in the abstract of [RFC8095], the IETF has standardized a number of protocols that provide transport services. Although these protocols, taken in total, provide a wide variety of transport services, Section 6 will distinguish between two extremes:

- transport protocols used to provide reliable, in-order media delivery to an endpoint, typically providing flow control and congestion control (Section 6.1), and
- transport protocols used to provide unreliable, unordered media delivery to an endpoint, without flow control or congestion control (Section 6.2).

Because newly standardized transport protocols, such as QUIC [RFC9000], that are typically implemented in user space can evolve their transport behavior more rapidly than currently used transport protocols that are typically implemented in operating system kernel space, this document includes a description of how the path characteristics that streaming media providers may see are likely to evolve; see Section 6.3.

It is worth noting explicitly that the transport protocol layer might include more than one protocol. For example, a specific media transport protocol might run over HTTP, or over WebTransport, which in turn runs over HTTP.

It is worth noting explicitly that more complex network protocol stacks are certainly possible -- for instance, when packets with this protocol stack are carried in a tunnel or in a VPN, the entire packet would likely appear in the payload of other protocols. If these environments are present, streaming media operators may need to analyze their effects on applications as well.

6.1. Media Transport over Reliable Transport Protocols

The HLS [RFC8216] and DASH [MPEG-DASH] media transport protocols are typically carried over HTTP, and HTTP has used TCP as its only standardized transport protocol until HTTP/3 [RFC9114]. These media transport protocols use ABR response strategies as described in Section 5 to respond to changing path characteristics, and underlying transport protocols are also attempting to respond to changing path characteristics.

The past success of the largely TCP-based Internet is evidence that the various flow control and congestion control mechanisms that TCP has used to achieve equilibrium quickly, at a point where TCP senders do not interfere with other TCP senders for sustained periods of time [RFC5681], have been largely successful. The Internet has continued to work even when the specific TCP mechanisms used to reach equilibrium changed over time [RFC7414]. Because TCP

provided a common tool to avoid contention, even when significant TCP-based applications like FTP were largely replaced by other significant TCP-based applications like HTTP, the transport behavior remained safe for the Internet.

Modern TCP implementations [RFC9293] continue to probe for available bandwidth and "back off" when a network path is saturated but may also work to avoid growing queues along network paths, which can prevent older TCP senders from quickly detecting when a network path is becoming saturated. Congestion control mechanisms, such as Copa [COPA18] and Bottleneck Bandwidth and Round-trip propagation time (BBR) [BBR-CONGESTION-CONTROL], make these decisions based on measured path delays, assuming that if the measured path delay is increasing, the sender is injecting packets onto the network path faster than the network can forward them (or the receiver can accept them), so the sender should adjust its sending rate accordingly.

Although common TCP behavior has changed significantly since the days of [Jacobson-Karels] and [RFC2001], even with adding new congestion controllers such as CUBIC [RFC8312], the common practice of implementing TCP as part of an operating system kernel has acted to limit how quickly TCP behavior can change. Even with the widespread use of automated operating system update installation on many end-user systems, streaming media providers could have a reasonable expectation that they could understand TCP transport protocol behaviors and that those behaviors would remain relatively stable in the short term.

6.2. Media Transport over Unreliable Transport Protocols

Because UDP does not provide any feedback mechanism to senders to help limit impacts on other users, UDP-based application-level protocols have been responsible for the decisions that TCP-based applications have delegated to TCP, i.e., what to send, how much to send, and when to send it. Because UDP itself has no transport-layer feedback mechanisms, UDP-based applications that send and receive substantial amounts of information are expected to provide their own feedback mechanisms and to respond to the feedback the application receives. This expectation is most recently codified as a Best Current Practice [RFC8085].

In contrast to adaptive segmented delivery over a reliable transport as described in Section 5.3, some applications deliver streaming media segments using an unreliable transport and rely on a variety of approaches, including:

- media encapsulated in a raw MPEG Transport Stream (MPEG-TS) [MPEG-TS] over UDP, which makes no attempt to account for reordering or loss in the transport,
- RTP [RFC3550], which can notice packet loss and repair some limited reordering,
- the Stream Control Transmission Protocol (SCTP) [RFC9260], which can use partial reliability [RFC3758] to recover from some loss but can abandon recovery to limit head-of-line blocking, and
- the Secure Reliable Transport (SRT) [SRT], which can use forward error correction and time-bound retransmission to recover from loss within certain limits but can abandon recovery to limit head-of-line blocking.

Under congestion and loss, approaches like the above generally experience transient media artifacts more often and delay of playback effects less often, as compared with reliable segment transport. Often, one of the key goals of using a UDP-based transport that allows some unreliability is to reduce latency and better support applications like videoconferencing or other live-action video with interactive components, such as some sporting events.

Congestion avoidance strategies for deployments using unreliable transport protocols vary widely in practice, ranging from being entirely unresponsive to responding by using strategies, including:

- feedback signaling to change encoder settings (as in [RFC5762]),
- fewer enhancement layers (as in [RFC6190]), and
- proprietary methods to detect QoE issues and turn off video to allow less bandwidth-intensive media, such as audio, to be delivered.

RTP relies on RTCP sender and receiver reports [RFC3550] as its own feedback mechanism and even includes circuit breakers for unicast RTP sessions [RFC8083] for situations when normal RTP congestion control has not been able to react sufficiently to RTP flows sending at rates that result in sustained packet loss.

The notion of "circuit breakers" has also been applied to other UDP applications in [RFC8084], such as tunneling packets over UDP that are potentially not congestion controlled (for example, "encapsulating MPLS in UDP", as described in [RFC7510]). If streaming media segments are carried in tunnels encapsulated in UDP, these media streams may encounter "tripped circuit breakers", with resulting user-visible impacts.

6.3. QUIC and Changing Transport Protocol Behavior

The QUIC protocol, developed from a proprietary protocol into an IETF Standards Track protocol [RFC9000], behaves differently than the transport protocols characterized in Sections 6.1 and 6.2.

Although QUIC provides an alternative to the TCP and UDP transport protocols, QUIC is itself encapsulated in UDP. As noted elsewhere in Section 7.1, the QUIC protocol encrypts almost all of its transport parameters and all of its payload, so any intermediaries that network operators may be using to troubleshoot HTTP streaming media performance issues, perform analytics, or even intercept exchanges in current applications will not work for QUIC-based applications without making changes to their networks. Section 7 describes the implications of media encryption in more detail.

While QUIC is designed as a general-purpose transport protocol and can carry different application-layer protocols, the current standardized mapping is for HTTP/3 [RFC9114], which describes how QUIC transport services are used for HTTP. The convention is for HTTP/3 to run over UDP port 443 [Port443], but this is not a strict requirement.

When HTTP/3 is encapsulated in QUIC, which is then encapsulated in UDP, streaming operators (and network operators) might see UDP traffic patterns that are similar to HTTP(S) over TCP. UDP ports may be blocked for any port numbers that are not commonly used, such as UDP 53 for DNS.

Even when UDP ports are not blocked and QUIC packets can flow, streaming operators (and network operators) may severely rate-limit this traffic because they do not expect to see legitimate high-bandwidth traffic, such as streaming media over the UDP ports that HTTP/3 is using.

As noted in Section 5.5.2, because TCP provides a reliable, in-order delivery service for applications, any packet loss for a TCP connection causes head-of-line blocking so that no TCP segments arriving after a packet is lost will be delivered to the receiving application until retransmission of the lost packet has been received, allowing in-order delivery to the application to continue. As described in [RFC9000], QUIC connections can carry multiple streams, and when packet losses do occur, only the streams carried in the lost packet are delayed.

A QUIC extension currently being specified [RFC9221] adds the capability for "unreliable" delivery, similar to the service provided by UDP, but these datagrams are still subject to the QUIC connection's congestion controller, providing some transport-level congestion avoidance measures, which UDP does not.

As noted in Section 6.1, there is an increasing interest in congestion control algorithms that respond to delay measurements instead of responding to packet loss. These algorithms may deliver an improved user experience, but in some cases, they have not responded to sustained packet loss, which exhausts available buffers along the end-to-end path that may affect other users sharing that path. The QUIC protocol provides a set of congestion control hooks that can be used for algorithm agility, and [RFC9002] defines a basic congestion control algorithm that is roughly similar to TCP NewReno [RFC6582]. However, QUIC senders can and do unilaterally choose to use different algorithms, such as loss-based CUBIC [RFC8312], delay-based Copa or BBR, or even something completely different.

The Internet community does have experience with deploying new congestion controllers without causing congestion collapse on the Internet. As noted in [RFC8312], both the CUBIC congestion controller and its predecessor BIC have significantly different behavior from Reno-style congestion controllers, such as TCP NewReno [RFC6582]; both were added to the Linux kernel to allow experimentation and analysis, both were then selected as the default TCP congestion controllers in Linux, and both were deployed globally.

The point mentioned in Section 6.1 about TCP congestion controllers being implemented in operating system kernels is different with QUIC. Although QUIC can be implemented in operating system kernels, one of the design goals when this work was chartered was "QUIC is expected to support rapid, distributed development and testing of features"; to meet this expectation, many implementers have chosen to implement QUIC in user space, outside the operating system kernel, and to even distribute QUIC libraries with their own applications. It is worth noting that streaming operators using HTTP/3, carried over QUIC, can expect more frequent deployment of new congestion controller behavior than has been the case with HTTP/1 and HTTP/2, carried over TCP.

It is worth considering that if TCP-based HTTP traffic and UDP-based HTTP/3 traffic are allowed to enter operator networks on roughly equal terms, questions of fairness and contention will be heavily dependent on interactions between the congestion controllers in use for TCP-based HTTP traffic and UDP-based HTTP/3 traffic.

7. Streaming Encrypted Media

"Encrypted Media" has at least three meanings:

- Media encrypted at the application layer, typically using some sort of Digital Rights Management (DRM) system or other object encryption/security mechanism and typically remaining encrypted at rest when senders and receivers store it.
- Media encrypted by the sender at the transport layer and remaining encrypted until it reaches the ultimate media consumer (in this document, it is referred to as end-to-end media encryption).
- Media encrypted by the sender at the transport layer and remaining encrypted until it reaches some intermediary that is *not* the ultimate media consumer but has credentials allowing decryption of the media content. This intermediary may examine and even transform the media content in some way, before forwarding re-encrypted media content (in this document, it is referred to as hop-by-hop media encryption).

This document focuses on media encrypted at the transport layer, whether encryption is performed hop by hop or end to end. Because media encrypted at the application layer will only be processed by application-level entities, this encryption does not have transport-layer implications. Of course, both hop-by-hop and end-to-end encrypted transport may carry media that is, in addition, encrypted at the application layer.

Each of these encryption strategies is intended to achieve a different goal. For instance, application-level encryption may be used for business purposes, such as avoiding piracy or enforcing geographic restrictions on playback, while transport-layer encryption may be used to prevent media stream manipulation or to protect manifests.

This document does not take a position on whether those goals are valid.

Both end-to-end and hop-by-hop media encryption have specific implications for streaming operators. These are described in Sections 7.2 and 7.3.

7.1. General Considerations for Streaming Media Encryption

The use of strong encryption does provide confidentiality for encrypted streaming media, from the sender to either the ultimate media consumer or to an intermediary that possesses credentials allowing decryption. This does prevent deep packet inspection (DPI) by any on-path intermediary that does not possess credentials allowing decryption. However, even encrypted content streams may be vulnerable to traffic analysis. An on-path observer that can identify that encrypted traffic contains a media stream could "fingerprint" this encrypted media stream and then compare it against "fingerprints" of known content. The protection provided by strong encryption can be further lessened if a streaming media operator is repeatedly encrypting the

same content. "Identifying HTTPS-Protected Netflix Videos in Real-Time" [CODASPY17] is an example of what is possible when identifying HTTPS-protected videos over TCP transport, based either on the length of entire resources being transferred or on characteristic packet patterns at the beginning of a resource being transferred. If traffic analysis is successful at identifying encrypted content and associating it with specific users, this tells an on-path observer what resource is being streamed, and by who, almost as certainly as examining decrypted traffic.

Because HTTPS has historically layered HTTP on top of TLS, which is in turn layered on top of TCP, intermediaries have historically had access to unencrypted TCP-level transport information, such as retransmissions, and some carriers exploited this information in attempts to improve transport-layer performance [RFC3135]. The most recent standardized version of HTTPS, HTTP/3 [RFC9114], uses the QUIC protocol [RFC9000] as its transport layer. QUIC relies on the TLS 1.3 initial handshake [RFC8446] only for key exchange [RFC9001] and encrypts almost all transport parameters itself, except for a few invariant header fields. In the QUIC short header, the only transport-level parameter that is sent "in the clear" is the Destination Connection ID [RFC8999], and even in the QUIC long header, the only transport-level parameters sent "in the clear" are the version, Destination Connection ID, and Source Connection ID. For these reasons, HTTP/3 is significantly more "opaque" than HTTPS with HTTP/1 or HTTP/2.

[RFC9312] discusses the manageability of the QUIC transport protocol that is used to encapsulate HTTP/3, focusing on the implications of QUIC's design and wire image on network operations involving QUIC traffic. It discusses what network operators can consider in some detail.

More broadly, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols" [RFC9065] describes the impact of increased encryption of transport headers in general terms.

It is also worth noting that considerations for heavily encrypted transport protocols also come into play when streaming media is carried over IP-level VPNs and tunnels, with the additional consideration that an intermediary that does not possess credentials allowing decryption will not have visibility to the source and destination IP addresses of the packets being carried inside the tunnel.

7.2. Considerations for Hop-by-Hop Media Encryption

Hop-by-hop media encryption offers the benefits described in Section 7.1 between the streaming media operator and authorized intermediaries, among authorized intermediaries, and between authorized intermediaries and the ultimate media consumer; however, it does not provide these benefits end to end. The streaming media operator and ultimate media consumer must trust the authorized intermediaries, and if these intermediaries cannot be trusted, the benefits of encryption are lost.

Although the IETF has put considerable emphasis on end-to-end streaming media encryption, there are still important use cases that require the insertion of intermediaries.

There are a variety of ways to involve intermediaries, and some are much more intrusive than others.

From a streaming media operator's perspective, a number of considerations are in play. The first question is likely whether the streaming media operator intends that intermediaries are explicitly addressed from endpoints or whether the streaming media operator is willing to allow intermediaries to "intercept" streaming content transparently, with no awareness or permission from either endpoint.

If a streaming media operator does not actively work to avoid interception by on-path intermediaries, the effect will be indistinguishable from "impersonation attacks", and endpoints cannot be assured of any level of confidentiality and cannot trust that the content received came from the expected sender.

Assuming that a streaming media operator does intend to allow intermediaries to participate in content streaming and does intend to provide some level of privacy for endpoints, there are a number of possible tools, either already available or still being specified. These include the following:

Server and Network Assisted DASH [MPEG-DASH-SAND]:
: This specification introduces explicit messaging between DASH clients and DASH-aware network elements or among various DASH-aware network elements for the purpose of improving the efficiency of streaming sessions by providing information about real-time operational characteristics of networks, servers, proxies, caches, CDNs, as well as a DASH client's performance and status.

"Double Encryption Procedures for the Secure Real-Time Transport Protocol (SRTP)" [RFC8723]:
: This specification provides a cryptographic transform for the SRTP that provides both hop-by-hop and end-to-end security guarantees.

Secure Frames [SFRAME]:
: [RFC8723] is closely tied to SRTP, and this close association impeded widespread deployment, because it could not be used for the most common media content delivery mechanisms. A more recent proposal, Secure Frames [SFRAME], also provides both hop-by-hop and end-to-end security guarantees but can be used with other media transport protocols beyond SRTP.

A streaming media operator's choice of whether to involve intermediaries requires careful consideration. As an example, when ABR manifests were commonly sent unencrypted, some access network operators would modify manifests during peak hours by removing high-bitrate renditions to prevent players from choosing those renditions, thus reducing the overall bandwidth consumed for delivering these media streams and thereby reducing the network load and improving the average user experience for their customers. Now that ubiquitous encryption typically prevents this kind of modification, a streaming media operator who used intermediaries in the past, and who now wishes to maintain the same level of network health and user experience, must choose between adding intermediaries who are authorized to change the manifests or adding some other form of complexity to their service.

Some resources that might inform other similar considerations are further discussed in [RFC8824] (for WebRTC) and [RFC9312] (for HTTP/3 and QUIC).

7.3. Considerations for End-to-End Media Encryption

End-to-end media encryption offers the benefits described in Section 7.1 from the streaming media operator to the ultimate media consumer.

End-to-end media encryption has become much more widespread in the years since the IETF issued "Pervasive Monitoring Is an Attack" [RFC7258] as a Best Current Practice, describing pervasive monitoring as a much greater threat than previously appreciated. After the Snowden disclosures, many content providers made the decision to use HTTPS protection -- HTTP over TLS -- for most or all content being delivered as a routine practice, rather than in exceptional cases for content that was considered sensitive.

However, as noted in [RFC7258], there is no way to prevent pervasive monitoring by an attacker while allowing monitoring by a more benign entity who only wants to use DPI to examine HTTP requests and responses to provide a better user experience. If a modern encrypted transport protocol is used for end-to-end media encryption, unauthorized on-path intermediaries are unable to examine transport and application protocol behavior. As described in Section 7.2, only an intermediary explicitly authorized by the streaming media operator who is to examine packet payloads, rather than intercepting packets and examining them without authorization, can continue these practices.

[RFC7258] states that "[t]he IETF will strive to produce specifications that mitigate pervasive monitoring attacks", so streaming operators should expect the IETF's direction toward preventing unauthorized monitoring of IETF protocols to continue for the foreseeable future.

8. Additional Resources for Streaming Media

The Media Operations (MOPS) community maintains a list of references and resources; for further reading, see [MOPS-RESOURCES].

9. IANA Considerations

This document has no IANA actions.

10. Security Considerations

Security is an important matter for streaming media applications, and the topic of media encryption was explained in Section 7. This document itself introduces no new security issues.

11. Informative References

[ABRSurvey]

Bentaleb, A., Taani, B., Begen, A. C., Timmerer, C., and R. Zimmermann, "A survey on bitrate adaptation schemes for streaming media over HTTP", IEEE Communications Surveys & Tutorials, vol. 21/1, pp. 562-585, Firstquarter 2019, DOI 10.1109/COMST.2018.2862938, <https://doi.org/10.1109/COMST.2018.2862938>.

[ADFRAUD] Sadeghpour, S. and N. Vlajic, "Ads and Fraud: A Comprehensive Survey of Fraud in Online Advertising", Journal of Cybersecurity and Privacy 1, no. 4, pp. 804-832, DOI 10.3390/jcp1040039, December 2021, <https://doi.org/10.3390/jcp1040039>.

[BALANCING] Berger, D., "Balancing Consumer Privacy with Behavioral Targeting", Santa Clara High Technology Law Journal, Vol. 27, Issue 1, Article 2, 2010, <https://digitalcommons.law.scu.edu/chtlj/vol27/iss1/2/>.

[BAP] Coalition for Better Ads, "Making Online Ads Better for Everyone", <https://www.betterads.org/>.

[BBR-CONGESTION-CONTROL] Cardwell, N., Cheng, Y., Yeganeh, S. H., Swett, I., and V. Jacobson, "BBR Congestion Control", Work in Progress, Internet-Draft, draft-cardwell-iccrg-bbr-congestion-control-02, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-02>.

[BEHAVE] Yan, J., Liu, N., Wang, G., Zhang, W., Jiang, Y., and Z. Chen, "How much can behavioral targeting help online advertising?", WWW '09: Proceedings of the 18th international conference on World wide web, pp. 261-270, DOI 10.1145/1526709.1526745, April 2009, <https://dl.acm.org/doi/abs/10.1145/1526709.1526745>.

[BEHAVE2] Goldfarb, A. and C. E. Tucker, "Online advertising, behavioral targeting, and privacy", Communications of the ACM, Volume 54, Issue 5, pp. 25-27, DOI 10.1145/1941487.1941498, May 2011, <https://dl.acm.org/doi/abs/10.1145/1941487.1941498>.

[CMAF-CTE] Bentaleb, A., Akcay, M., Lim, M., Begen, A., and R. Zimmermann, "Catching the Moment With LoL+ in Twitch-Like Low-Latency Live Streaming Platforms", IEEE Trans. Multimedia, Vol. 24, pp. 2300-2314, DOI 10.1109/TMM.2021.3079288, May 2021, <https://doi.org/10.1109/TMM.2021.3079288>.

[CODASPY17] Reed, A. and M. Kranch, "Identifying HTTPS-Protected Netflix Videos in Real-Time", ACM CODASPY, DOI 10.1145/3029806.3029821, March 2017, <https://dl.acm.org/doi/10.1145/3029806.3029821>.

[CoDel] Nichols, K. and V. Jacobson, "Controlling queue delay", Communications of the ACM, Volume 55, Issue 7, pp. 42-50", DOI 10.1145/2209249.2209264, July 2012, <https://doi.org/10.1145/2209249.2209264>.

[COPA18] Arun, V. and H. Balakrishnan, "Copa: Practical Delay-Based Congestion Control for the Internet", USENIX NSDI, April 2018, <https://web.mit.edu/copa/>.

[CTA-2066] Consumer Technology Association, "Streaming Quality of Experience Events, Properties and Metrics", CTA-2066, March 2020, <https://shop.cta.tech/products/streaming-quality-of-experience-events-properties-and-metrics>.

[CTA-5004] Consumer Technology Association, "Web Application Video Ecosystem - Common Media Client Data", CTA-5004, September 2020, <https://shop.cta.tech/products/web-application-video-ecosystem-common-media-client-data-cta-5004>.

[CVNI] Cisco, "Cisco Visual Networking Index: Forecast and Trends, 2017–2022", 2018.

[ELASTIC] De Cicco, L., Caldaralo, V., Palmisano, V., and S. Mascolo, "ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH)", Packet Video Workshop, DOI 10.1109/PV.2013.6691442, December 2013, <https://ieeexplore.ieee.org/document/6691442>.

[Encodings] Apple Developer, "HTTP Live Streaming (HLS) Authoring Specification for Apple Devices", June 2020, <https://developer.apple.com/documentation/http_live_streaming/hls_authoring_specification_for_apple_devices>.

[HLS-RFC8216BIS] Pantos, R., Ed., "HTTP Live Streaming 2nd Edition", Work in Progress, Internet-Draft, draft-pantos-hls-rfc8216bis-11, 11 May 2022, <https://www.ietf.org/archive/id/draft-pantos-hls-rfc8216bis-11.txt>.

[IAB-ADS] "IAB", <https://www.iab.com/>.

[ISOBMFF] ISO, "Information technology - Coding of audio-visual objects - Part 12: ISO base media file format", ISO/IEC 14496-12:2022, January 2022, <https://www.iso.org/standard/83102.html>.

[Jacobson-Karels] Jacobson, V. and M. Karels, "Congestion Avoidance and Control", November 1988, <https://ee.lbl.gov/papers/congavoid.pdf>.

[LL-DASH] DASH-IF, "Low-latency Modes for DASH", March 2020, <https://dashif.org/docs/CR-Low-Latency-Live-r8.pdf>.

[Micro] Taher, T. M., Misurac, M. J., LoCicero, J. L., and D. R. Ucci, "Microwave Oven Signal Interference Mitigation For Wi-Fi Communication Systems", 2008 5th IEEE Consumer Communications and Networking Conference, pp. 67-68, DOI 10.1109/ccnc08.2007.21, January 2008, <https://doi.org/10.1109/ccnc08.2007.21>.

[MMSP20] Durak, K. et al., "Evaluating the Performance of Apple's Low-Latency HLS", IEEE MMSP, DOI 10.1109/MMSP48831.2020.9287117, September 2020, <https://ieeexplore.ieee.org/document/9287117>.

[MMSys11] Akhshabi, S., Begen, A. C., and C. Dovrolis, "An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP", ACM MMSys, DOI 10.1145/1943552.1943574, February 2011, <https://dl.acm.org/doi/10.1145/1943552.1943574>.

[MOPS-RESOURCES] "rfc9317-additional-resources", September 2022, <https://wiki.ietf.org/group/mops/rfc9317-additional-resources>.

[MPEG-CMAF] ISO, "Information technology - Multimedia application format (MPEG-A) - Part 19: Common media application format (CMAF) for segmented media", ISO/IEC 23000-19:2020, March 2020, <https://www.iso.org/standard/79106.html>.

[MPEG-DASH] ISO, "Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats", ISO/IEC 23009-1:2022, August 2022, <https://www.iso.org/standard/83314.html>.

[MPEG-DASH-SAND] ISO, "Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 5: Server and network assisted DASH (SAND)", ISO/IEC 23009-5:2017, February 2017, <https://www.iso.org/standard/69079.html>.

[MPEG-TS] ITU-T, "Information technology - Generic coding of moving pictures and associated audio information: Systems", ITU-T Recommendation H.222.0, June 2021, <https://www.itu.int/rec/T-REC-H.222.0>.

[MPEGI] Boyce, J. M. et al., "MPEG Immersive Video Coding Standard", Proceedings of the IEEE, Vol. 109, Issue 9, pp. 1521-1536, DOI 10.1109/JPROC.2021.3062590, <https://ieeexplore.ieee.org/document/9374648>.

[OReilly-HPBN] Grigorik, I., "High Performance Browser Networking - Chapter 2: Building Blocks of TCP", May 2021, <https://hpbn.co/building-blocks-of-tcp/>.

[PCC] Schwarz, S. et al., "Emerging MPEG Standards for Point Cloud Compression", IEEE Journal on Emerging and Selected Topics in Circuits and Systems, DOI 10.1109/JETCAS.2018.2885981, March 2019, <https://ieeexplore.ieee.org/document/8571288>.

[Port443] IANA, "Service Name and Transport Protocol Port Number Registry", <https://www.iana.org/assignments/service-names-port-numbers>.

[RFC2001] Stevens, W., "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms", RFC 2001, DOI 10.17487/RFC2001, January 1997, <https://www.rfc-editor.org/info/rfc2001>.

[RFC2736] Handley, M. and C. Perkins, "Guidelines for Writers of RTP Payload Format Specifications", BCP 36, RFC 2736, DOI 10.17487/RFC2736, December 1999, <https://www.rfc-editor.org/info/rfc2736>.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <https://www.rfc-editor.org/info/rfc3135>.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <https://www.rfc-editor.org/info/rfc3168>.

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

[RFC3758] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Partial Reliability Extension", RFC 3758, DOI 10.17487/RFC3758, May 2004, <https://www.rfc-editor.org/info/rfc3758>.

[RFC4733] Schulzrinne, H. and T. Taylor, "RTP Payload for DTMF Digits, Telephony Tones, and Telephony Signals", RFC 4733, DOI 10.17487/RFC4733, December 2006, <https://www.rfc-editor.org/info/rfc4733>.

[RFC5481] Morton, A. and B. Claise, "Packet Delay Variation Applicability Statement", RFC 5481, DOI 10.17487/RFC5481, March 2009, <https://www.rfc-editor.org/info/rfc5481>.

[RFC5594] Peterson, J. and A. Cooper, "Report from the IETF Workshop on Peer-to-Peer (P2P) Infrastructure, May 28, 2008", RFC 5594, DOI 10.17487/RFC5594, July 2009, <https://www.rfc-editor.org/info/rfc5594>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC5762] Perkins, C., "RTP and the Datagram Congestion Control Protocol (DCCP)", RFC 5762, DOI 10.17487/RFC5762, April 2010, <https://www.rfc-editor.org/info/rfc5762>.

[RFC6190] Wenger, S., Wang, Y.-K., Schierl, T., and A. Eleftheriadis, "RTP Payload Format for Scalable Video Coding", RFC 6190, DOI 10.17487/RFC6190, May 2011, <https://www.rfc-editor.org/info/rfc6190>.

[RFC6582] Henderson, T., Floyd, S., Gurtov, A., and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC 6582, DOI 10.17487/RFC6582, April 2012, <https://www.rfc-editor.org/info/rfc6582>.

[RFC6817] Shalunov, S., Hazel, G., Iyengar, J., and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)", RFC 6817, DOI 10.17487/RFC6817, December 2012, <https://www.rfc-editor.org/info/rfc6817>.

[RFC6843] Clark, A., Gross, K., and Q. Wu, "RTP Control Protocol (RTCP) Extended Report (XR) Block for Delay Metric Reporting", RFC 6843, DOI 10.17487/RFC6843, January 2013, <https://www.rfc-editor.org/info/rfc6843>.

[RFC7258] Farrell, S. and H. Tschofenig, "Pervasive Monitoring Is an Attack", BCP 188, RFC 7258, DOI 10.17487/RFC7258, May 2014, <https://www.rfc-editor.org/info/rfc7258>.

[RFC7414] Duke, M., Braden, R., Eddy, W., Blanton, E., and A. Zimmermann, "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 7414, DOI 10.17487/RFC7414, February 2015, <https://www.rfc-editor.org/info/rfc7414>.

[RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <https://www.rfc-editor.org/info/rfc7510>.

[RFC7656] Lennox, J., Gross, K., Nandakumar, S., Salgueiro, G., and B. Burman, Ed., "A Taxonomy of Semantics and Mechanisms for Real-Time Transport Protocol (RTP) Sources", RFC 7656, DOI 10.17487/RFC7656, November 2015, <https://www.rfc-editor.org/info/rfc7656>.

[RFC7661] Fairhurst, G., Sathiaseelan, A., and R. Secchi, "Updating TCP to Support Rate-Limited Traffic", RFC 7661, DOI 10.17487/RFC7661, October 2015, <https://www.rfc-editor.org/info/rfc7661>.

[RFC8083] Perkins, C. and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions", RFC 8083, DOI 10.17487/RFC8083, March 2017, <https://www.rfc-editor.org/info/rfc8083>.

[RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <https://www.rfc-editor.org/info/rfc8084>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085, March 2017, <https://www.rfc-editor.org/info/rfc8085>.

[RFC8095] Fairhurst, G., Ed., Trammell, B., Ed., and M. Kuehlewind, Ed., "Services Provided by IETF Transport Protocols and Congestion Control Mechanisms", RFC 8095, DOI 10.17487/RFC8095, March 2017, <https://www.rfc-editor.org/info/rfc8095>.

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

[RFC8312] Rhee, I., Xu, L., Ha, S., Zimmermann, A., Eggert, L., and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks", RFC 8312, DOI 10.17487/RFC8312, February 2018, <https://www.rfc-editor.org/info/rfc8312>.

[RFC8404] Moriarty, K., Ed. and A. Morton, Ed., "Effects of Pervasive Encryption on Operators", RFC 8404, DOI 10.17487/RFC8404, July 2018, <https://www.rfc-editor.org/info/rfc8404>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8622] Bless, R., "A Lower-Effort Per-Hop Behavior (LE PHB) for Differentiated Services", RFC 8622, DOI 10.17487/RFC8622, June 2019, <https://www.rfc-editor.org/info/rfc8622>.

[RFC8723] Jennings, C., Jones, P., Barnes, R., and A.B. Roach, "Double Encryption Procedures for the Secure Real-Time Transport Protocol (SRTP)", RFC 8723, DOI 10.17487/RFC8723, April 2020, <https://www.rfc-editor.org/info/rfc8723>.

[RFC8824] Minaburo, A., Toutain, L., and R. Andreasen, "Static Context Header Compression (SCHC) for the Constrained Application Protocol (CoAP)", RFC 8824, DOI 10.17487/RFC8824, June 2021, <https://www.rfc-editor.org/info/rfc8824>.

[RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, January 2021, <https://www.rfc-editor.org/info/rfc8825>.

[RFC8834] Perkins, C., Westerlund, M., and J. Ott, "Media Transport and Use of RTP in WebRTC", RFC 8834, DOI 10.17487/RFC8834, January 2021, <https://www.rfc-editor.org/info/rfc8834>.

[RFC8835] Alvestrand, H., "Transports for WebRTC", RFC 8835, DOI 10.17487/RFC8835, January 2021, <https://www.rfc-editor.org/info/rfc8835>.

[RFC8999] Thomson, M., "Version-Independent Properties of QUIC", RFC 8999, DOI 10.17487/RFC8999, May 2021, <https://www.rfc-editor.org/info/rfc8999>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/info/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/info/rfc9001>.

[RFC9002] Iyengar, J., Ed. and I. Swett, Ed., "QUIC Loss Detection and Congestion Control", RFC 9002, DOI 10.17487/RFC9002, May 2021, <https://www.rfc-editor.org/info/rfc9002>.

[RFC9065] Fairhurst, G. and C. Perkins, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols", RFC 9065, DOI 10.17487/RFC9065, July 2021, <https://www.rfc-editor.org/info/rfc9065>.

[RFC9075] Arkko, J., Farrell, S., Kühlewind, M., and C. Perkins, "Report from the IAB COVID-19 Network Impacts Workshop 2020", RFC 9075, DOI 10.17487/RFC9075, July 2021, <https://www.rfc-editor.org/info/rfc9075>.

[RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/info/rfc9111>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/info/rfc9114>.

[RFC9221] Pauly, T., Kinnear, E., and D. Schinazi, "An Unreliable Datagram Extension to QUIC", RFC 9221, DOI 10.17487/RFC9221, March 2022, <https://www.rfc-editor.org/info/rfc9221>.

[RFC9260] Stewart, R., Tüxen, M., and K. Nielsen, "Stream Control Transmission Protocol", RFC 9260, DOI 10.17487/RFC9260, June 2022, <https://www.rfc-editor.org/info/rfc9260>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[RFC9312] Kühlewind, M. and B. Trammell, "Manageability of the QUIC Transport Protocol", RFC 9312, DOI 10.17487/RFC9312, September 2022, <https://www.rfc-editor.org/info/rfc9312>.

[SFRAME] IETF, "Secure Frame (sframe)", <https://datatracker.ietf.org/doc/draft-ietf-sframe-enc/>.

[SRT] Sharabayko, M., "SRT Protocol Overview", April 2020, <https://datatracker.ietf.org/meeting/interim-2020-mops-01/materials/slides-interim-2020-mops-01-sessa-srt-protocol-overview-00>.

[Survey360] Yaqoob, A., Bi, T., and G. Muntean, "A Survey on Adaptive 360° Video Streaming: Solutions, Challenges and Opportunities", IEEE Communications Surveys & Tutorials, Volume 22, Issue 4, DOI 10.1109/COMST.2020.3006999, July 2020, <https://ieeexplore.ieee.org/document/9133103>.

Acknowledgments

Thanks to Nancy Cam-Winget, Leslie Daigle, Roman Danyliw, Glenn Deen, Martin Duke, Linda Dunbar, Lars Eggert, Mike English, Roni Even, Aaron Falk, Alexandre Gouaillard, Erik Kline, Renan Krishna, Warren Kumari, Will Law, Chris Lemmons, Kiran Makhjani, Sanjay Mishra, Mark Nottingham, Dave Oran, Lucas Pardue, Tommy Pauly, Kyle Rose, Zahed Sarker, Michael Scharf, John Scudder, Valery Smyslov, Matt Stock, Éric Vyncke, and Robert Wilton for very helpful suggestions, reviews, and comments.

Authors' Addresses

Jake Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America
Email: jakeholland.net@gmail.com

Ali Begen
Networked Media
Turkey
Email: ali.begen@networked.media

Spencer Dawkins
Tencent America LLC
United States of America
Email: spencerdawkins.ietf@gmail.com