WOWZA media systems

LIVE-STREAMING VIDEO

LATENCY SCENARIOS FOR EVERY USE CASE

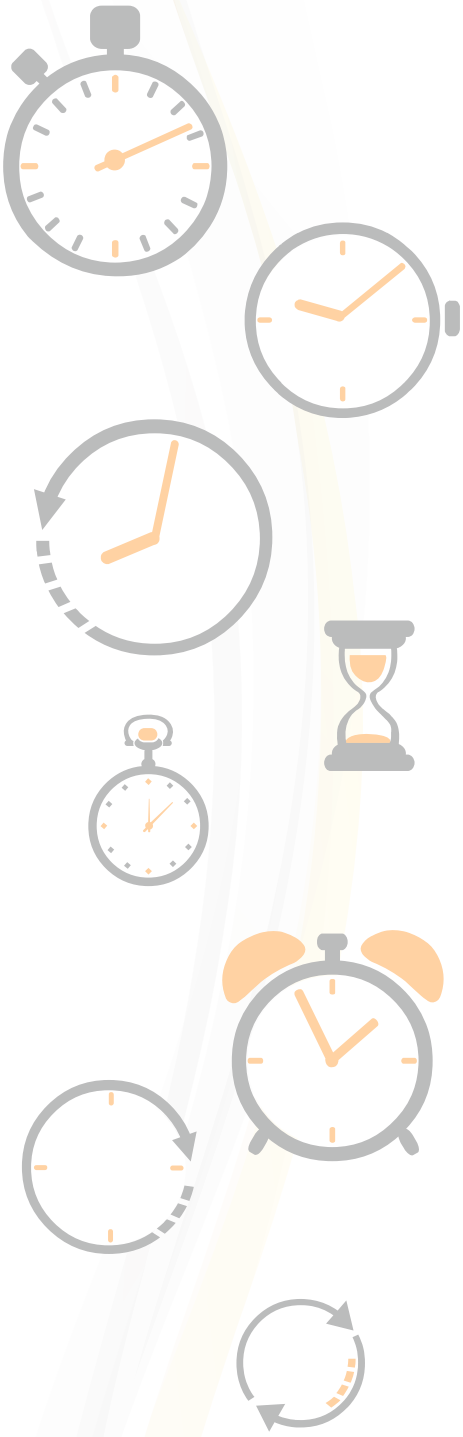
LOW LATENCY

ALL-CONDITIONS PLAYBACK

HIGH RESOLUTION

INTRODUCTION

Live-streaming video is now ubiquitous across industries. Whether it's news, sports, game-streaming services, user-generated content (UGC) apps or telepresence and Web conferencing solutions, everyone wants a piece of the real-time action. But despite nearly two decades of video-streaming refinement, one complaint is still common from video streamers and viewers: latency is too high.



Latency refers to the delay between the time a live stream is captured on camera and the time it appears on a remote viewer's screen. Live-streaming content can result in a significant amount of delay, at least when compared to over-the-air (OTA) or cable television broadcasts. Even though "live" cable broadcasts have five to 10 seconds of latency, viewers perceive the broadcast signal as being instantly available, in real time, when they turn on their TV—and they expect their live-streaming video to perform the same way.

For many use cases, streaming at or near real time is crucial to the user experience (UX). For example, live sports streams must keep up with cable and satellite broadcasts, so the game-winning play isn't spoiled for streamers by friends, family and social-network users watching on TV. Game streamers need real-time input from viewers on how to beat an opponent. Live-streaming news apps need to keep pace with network and cable TV broadcasts when breaking stories. And for telepresence and video conferencing, low latency is key for simulating real-life interaction.

However, this presents a challenge for the streaming industry: It must strike a balance between traditional streaming protocols and specialized players that provide relatively low latencies, on one hand, and Web-friendly, massively scalable streaming formats offering longer latencies on the other. Using HTTP-based streaming with HTML5 video players that don't require additional plugins is an appealing option—but the higher latencies that typically come with them are less desirable.

What's more, due to recent decisions by major technology providers such as Apple and Google, universal support for Flash Player in browsers is fading fast. For many years, the Flash browser plugin has led the pack in delivering low-latency audio and video streams across markets. So, what are the alternatives?

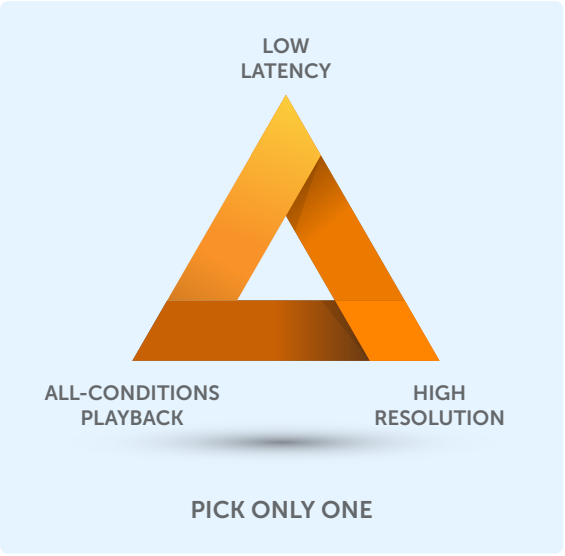
Luckily, there are several options for broadcasters and content providers, depending on the intended use case. In this guide, we'll discuss:

- Latency causes and considerations
- Streaming delivery methods
- Emerging low-latency technologies, such as WebRTC and WebSocket

YOU CAN'T HAVE IT ALL: CONSTRAINTS ON STREAMING VIDEO

For our purposes, video streaming has three conflicting constraints when it comes to playback quality and interactivity:

- **Low latency**
- **High resolution**
- **All-conditions playback** (the ability to play, without interruption, on any device under a variety of network and playback conditions)



Latency is a matter of time: The greater the degree of interactivity required, the lower latency must be.

In most use cases, you can effectively meet two of these constraints—but achieving all three is really challenging. Why? These conditions tend to work against each other: If you want to play at high resolution on the maximum number of devices, latency will usually be higher ; if you want to stream at low latency to a large audience, quality tends to suffer; and so on.

The ability to stream in any network condition is a reason many have adopted adaptive bitrate (ABR) playback methods. This allows for multiple bitrates (i.e., high quality) to be delivered, and the user, or playback software, to adjust the stream based on available bandwidth. All-conditions playback is also. All-conditions playback is also massively scalable through global content delivery networks (CDNs). However, many of the ABR-friendly protocols inject latency in order to have stable streams to the edge, and CDNs add latency of their own. Thus, optimizing for all-conditions playback at scale generally comes at the expense of low latency.

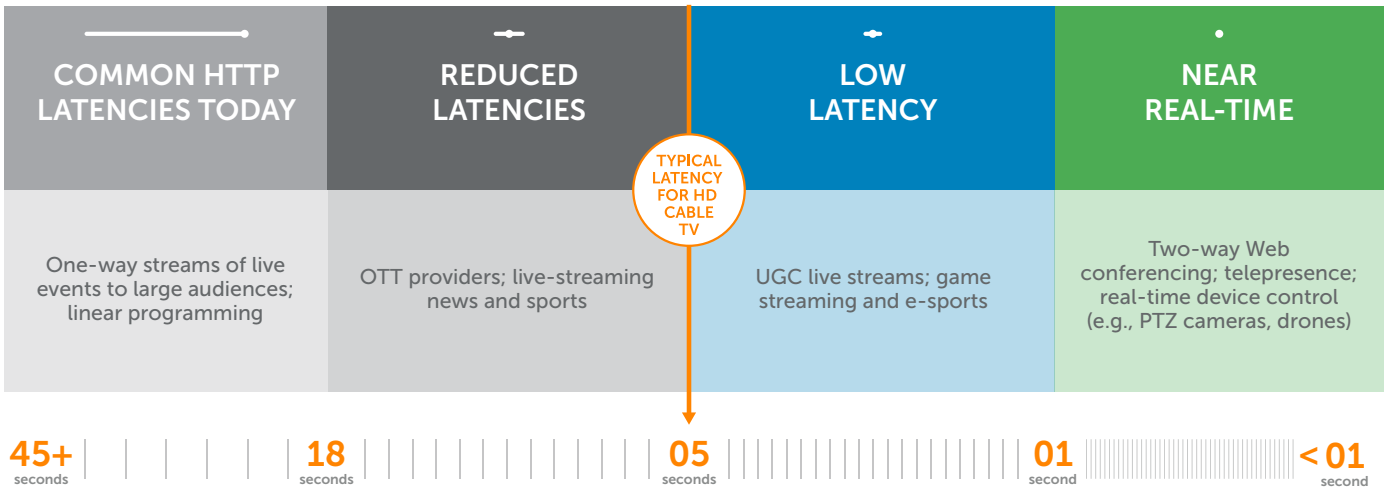
For any given deployment, a variety of factors at both ends of the acquisition and playback chain can contribute to latency, including:

- Encoding and/or transcoding settings
- Distribution methods
- Network topologies
- The video capture workflow (even at a camera level)
- The players and hardware being used to view live streams

However, video-streaming constraints have noticeable impacts on the end-user experience in a variety of use cases. The higher the degree of interactivity involved, the lower-latency streams you must deliver.

WOWZA media systems

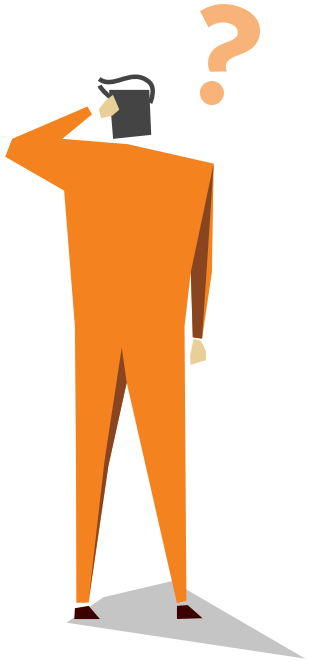
STREAMING LATENCY AND INTERACTIVITY CONTINUUM



Higher degrees of interactivity require lower levels of latency to provide a positive UX. Choose the latency scenario and delivery option that fits your use case.

WHAT DELIVERY OPTION SHOULD YOU CHOOSE?

Based on your use case and corresponding latency requirements, you can choose an appropriate encoding format and streaming protocol. Again, the degree of latency also impacts the level of viewer-to-broadcaster interactivity you're likely to achieve (if any). Understanding which protocols are designed for low latency, how they're deployed and their lowest latency thresholds will help you make more informed design decisions.



There are a variety of common streaming protocols, each with their own ranges of possible latency: from under 100 milliseconds to over 45 seconds. These protocols fall into three main categories:

- Traditional streaming protocols (RTSP, RTMP)
- HTTP-based adaptive streaming protocols (HLS, HDS, Smooth Streaming, MPEG-DASH)
- Emerging protocols (WebRTC)

Now, we'll look at each protocol category and which use cases they're best for.

TRADITIONAL STREAMING PROTOCOLS

The two most popular traditional streaming protocols are:

RTSP (Real-Time Streaming Protocol):
An industry standard for two decades, RTSP has been used for low-latency audio and video. It was used heavily by Apple, RealNetworks and Microsoft in their past streaming products.

RTMP (Real-Time Messaging Protocol):
Another industry staple, created to transfer audio, video and data with Adobe Flash applications.

RTMP and RTSP streaming has historically worked well for small to midsize audiences from a single media server. Latencies vary from a few hundred milliseconds to multiple seconds, depending on use case, configuration and architecture.

Broader reach with these protocols requires many media servers, some repeating or reflecting streams from the main (origin) server. This linear scaling of additional media servers for larger audiences, either in a tiered or edge network configuration, ends up making the per-stream delivery cost rise significantly.

Traditional Streaming Protocols

Common Protocols: *RTSP, RTMP*

Latency Range: *Low; a few hundred milliseconds to many seconds.*

Use Cases: *Streaming for small to midsize audiences from a single media server; audio and video conferencing.*

Constraint Performance: *Low latency achieved when high resolution is sacrificed.*

Scalability: *Requires multiple media servers, which can be costly.*

Native Players? *No; must use a dedicated media player app or browser plug-in (e.g., Adobe Flash Player).*

So, how do these protocols perform under our quality and interactivity constraints? Especially for low-latency situations, neither protocol is designed to adapt to changing network conditions. To ensure low-latency delivery across spotty network connections, you might need to reduce bandwidth (and thus give up high resolution) by delivering a stream with lowest-common-denominator quality; otherwise, you risk interruptions during playback.

In addition, RTSP and RTMP are not natively supported on many endpoints. Their use requires a dedicated media player app or a browser plug-in, such as Adobe Flash Player for RTMP streams. However, as mentioned, Apple and Google have both started phasing out support for Flash Player in their respective Safari and Chrome browsers.

Finally, traditional streaming protocols such as RTMP and RTSP require opening up ports beyond the standard HTTP port 80—and these additional ports are commonly restricted by firewalls for security reasons. This, combined with scaling challenges and lack of support on endpoints, shrinks the overall audience that can view content streamed by RTMP or RTSP.

HTTP-BASED ADAPTIVE STREAMING PROTOCOLS

HTTP-Based Adaptive Streaming Protocols

Common Protocols: *HLS, HDS, HSS, MPEG-DASH*

Latency Range: *Default HTTP latency (18-45 seconds)*

Use Cases: *Live streaming for large audiences on a wide range of devices, including mobile*

Constraint Performance: *All-conditions playback and high resolution in exchange for higher latency*

Scalability: *Can be done from a single media server using CDNs; cost-effective*

Native Players? *Yes (e.g., Apple QuickTime, native iOS and Android players, Microsoft Silverlight)*

These newer audio and video delivery protocols take advantage of the widely used HTTP port 80, and deliver streaming content as a series of "chunks." In essence, this HTTP-based approach sends hundreds or thousands of small, cacheable files in sequence, which are then reassembled into an audio and video stream. The stream is often played using a specialized player. However, most modern browsers now support extended HTML5 media elements, which can also be used to play streams.

One of the major upsides of HTTP-based delivery is the ability to scale to large audiences on a range of devices—from laptops and desktops to smartphones and tablets—without needing to scale out, manage and pay for media servers. Often, a single media server can act as an origin server for a large group of HTTP caching and delivery servers, such as those found in a CDN.

Another benefit is that these protocols offer adaptive bitrate (ABR) streaming. ABR streaming involves simultaneously sending HTTP chunks with varying resolutions and bitrates. A manifest file tells the player which quality options are available—including lower-bandwidth resolutions for mobile networks, and higher resolutions for Wi-Fi or wired Internet connections. The player logic can then select the highest possible quality that can be supported at any given moment, adapting as conditions change.

There are a variety of HTTP-based ABR protocols, backed by different providers:

- Adobe HDS (HTTP Dynamic Streaming)
- Apple HLS (HTTP Live Streaming)
- Microsoft HSS (HTTP Smooth Streaming)
- MPEG-DASH (Moving Picture Experts Group - Dynamic Adaptive Streaming over HTTP)—which all three companies collaborated to create, and is the only ABR format ratified as an international standard

Based on what we hear from content distributors, usage of HDS and HSS is likely to fade over time. Due largely to the Apple iTunes requirement to use HLS for iOS streaming, non-standardized HLS has the lion's share of the market among ABR formats today. With the international standardization of DASH and its adoption by many other consumer electronic standards groups, it seemed likely to challenge HLS dominance—however, questions about possible DASH royalties have slowed its adoption.

HTTP-BASED ADAPTIVE STREAMING PROTOCOLS (CONTINUED)

A new Common Media Application Format (CMAF) standard is now emerging that will soon make either HLS or DASH a safe bet. CMAF chunks will allow you to use the same HTTP audio and video for both DASH and HLS, changing only the manifest used by the player to retrieve the chunks. What's more, with Android adoption in Honeycomb, HLS has become ubiquitous in mobile.

Each of these ABR formats provides excellent scalability using standard CDNs. However, latency ranges from 15 to 45 seconds. Looking at our quality constraints, HTTP-based ABR protocols facilitate all-conditions playback for everyone at the highest resolution their device and network connection can sustain—but this comes at the expense of low latency. And again, CDNs inject additional latency.

One common question is: Can the HTTP chunks, which are typically between two and 10 seconds long, be shortened to one-second lengths to make latency competitive with cable TV? The short answer is, yes and no.

An HTTP-based protocol can be tuned for low-latency delivery at closer to five seconds; in fact, Facebook's custom infrastructure uses MPEG-DASH with one-second chunk sizes, resulting in less than three seconds of latency. However, Facebook's is one of the most complex use cases in the industry. For most purposes, chunks cannot practically be lowered below one second—at least, not without introducing network inefficiencies and likely glitches for viewers.

Since they can't be lowered to near real-time delivery, there's no practical way to use HTTP-based delivery for applications where the UX relies on instant interactivity, such as video chat. However, these protocols are great for broadcasting live events to large audiences viewing low-latency or time-shifted content across a wide range of modern, ABR-ready devices.

EMERGING TECHNOLOGY

While HTTP-based formats are great for scalability, they don't work for real-time applications; so, what options do we have for real-time streaming? Luckily, there are two up-and-coming technologies: WebSocket and WebRTC.

WebSocket is a communications protocol that can be used with other streaming protocols, such as RTMP, WebRTC or other commercially available protocols, including Haivision SRT, Wowza WOWZ and Aspera FASP.

WebSocket was designed to provide a standardized, two-way, reliable communications channel between a browser and a server. It's often used to send text, metadata and other information with latencies of under 200 milliseconds, even on older cellular networks. If data gets lost in transmission, it will wait for the data to be re-sent and successfully arrive. That's great when connections are stable—but when they are unreliable, latency can be introduced, and streams may be interrupted when the lost information is requested again.

One of the newest entrants for streaming is WebRTC (Web Real-Time Communications). As its name states, WebRTC is

Emerging Technology

Common Protocols: *WebSocket, WebRTC*

Latency Range: *Real-time; one second or less*

Use Cases: *Real-time live streaming, chat and video conferencing*

Constraint Performance: *Low latency, high resolution and a partial win for all-conditions playback*

Scalability: *Can be scaled up from media or edge servers without specialized plug-in architectures*

Native Players? *Not necessary; native stream encoding and playback in compatible browsers available*

EMERGING TECHNOLOGY (CONTINUED)

designed for real-time audio, video and data delivery over less-reliable connections. It's often referred to as a protocol, as we do here, but it's really a collection of protocols and APIs—and is in the process of being ratified as a standard. While not as simple as WebSocket, it has a greater depth of features, and its adoption has been championed by Google.

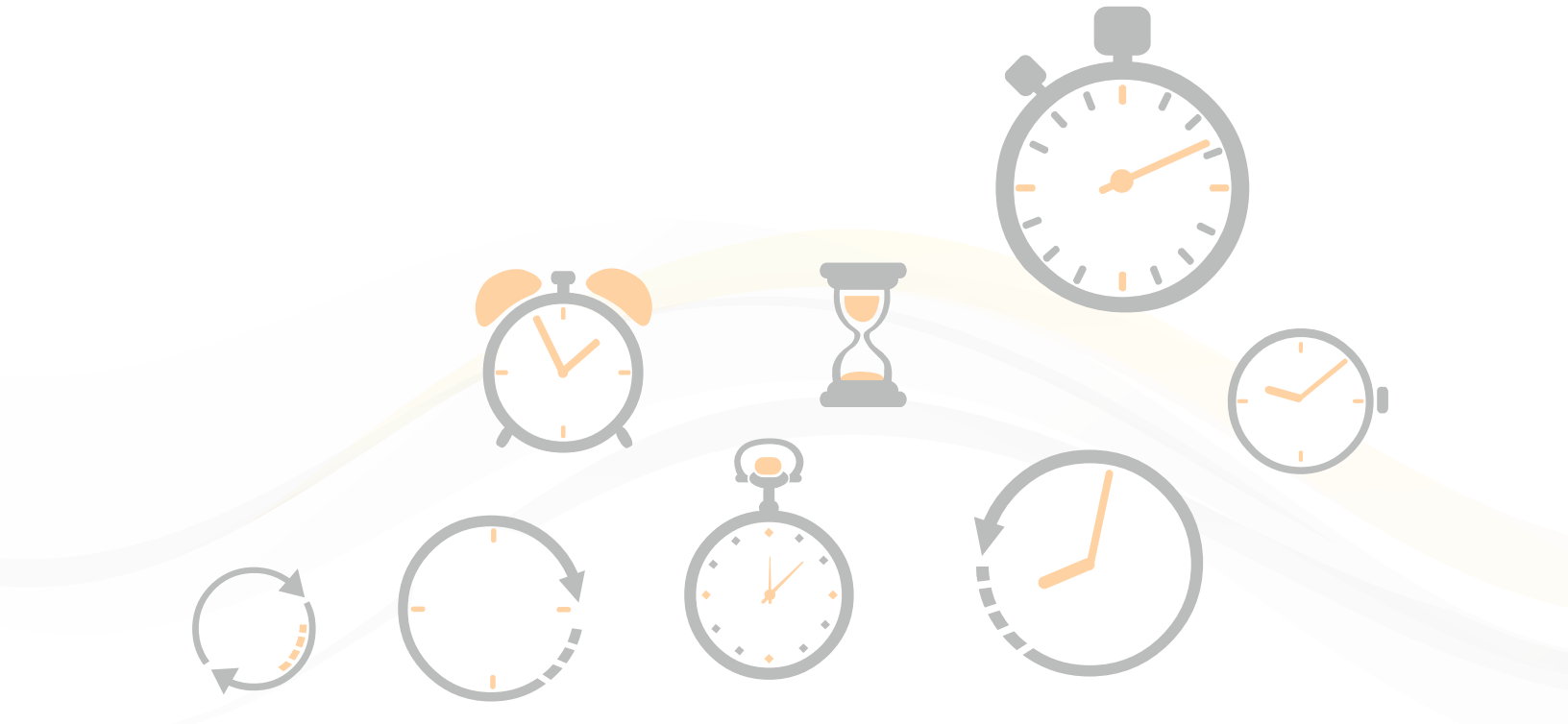
WebRTC applications are easily deployed without plug-ins, thanks in part to native stream encoding and playback available in compatible HTML5 browsers (Safari is the exception, but Apple has hinted WebRTC support is coming). WebRTC accommodates latencies of one second or less, and as low as 200 milliseconds, on par with traditional video conferencing hardware endpoints.

Like video conferencing hardware, WebRTC can use VP8/H.264 for the primary codec, so it's perfectly suited for one-to-one or small-group live chat. WebRTC is also popular for creating browser-based streaming applications; it's best supported by Chrome, Firefox, Microsoft Edge or Microsoft Internet Explorer.

However, the true appeal of WebRTC is that it can be scaled up, using more robust media servers as well as edge servers. In this way, streaming architectures are similar to RTMP or RTSP delivery approaches—without the need for specialized plug-in architectures at the endpoints.

It takes more work to scale WebSocket and WebRTC streaming than HTTP streaming. But the low-latency reach of these emerging protocols to most modern HTML5 browsers helps bridge the gap between protocols such as RTMP and HLS. In this way, emerging protocols perform best under our streaming constraints: low latency and high resolution are achieved, and all-conditions playback is largely accomplished. While you don't get adaptive streaming, you can reach most screens.

In some applications, WebSocket and WebRTC can be used together. For example, WebSocket can be used to set up connections between two parties, with WebRTC handling to exchange audio and video. With their direct-to-browser communications, both the WebRTC and WebSocket protocols can provide real-time communications capabilities—and help replace older protocols.



WOWZA media systems

THE FOUR LATENCY SCENARIOS: WHAT'S YOUR USE CASE?

Two-Way and Chat

Required Latency: Real-time streaming (<1 second)

Use Cases: Two-way Web conferencing; telepresence; real-time device control (e.g., PTZ cameras, drones)

Interactivity Level: High

User Experience: Many-to-many; many-to-one (e.g., multiple game streamers synced to the same display); one-to-one communication (e.g., telepresence)

Best Fit: WebRTC protocol, or WebSocket with RTMP

Keep in Mind: When broadcasting one-to-many streams, may need to use a single server for a given event. When synchronizing a real-time video stream and the output of a multi-camera video mixer, may need to use traditional SDI-based video mixing around for your image magnification (IMAG) and multi-display video-wall technology solutions.

Interactive

Required Latency: Low (1-5 seconds)

Use Cases: UGC live streams; game streaming and e-sports

Interactivity Level: Medium

User Experience: Optional viewer-to-viewer or viewer-to-broadcaster engagement (e.g., apps with ephemeral video; live event broadcasts on UGC or social media platforms, such as awards shows, protests or elections)

Best Fit: Tuned HTTP ABR streaming using RTMP, RTSP and/or WebSocket; WebRTC protocol with applications/browsers tuned to enable origin routing and eliminate latency bottlenecks

Keep in Mind: You may need to load-balance and/or have a robust edge network when using tuned HTTP ABR streaming. Encoding options for WebRTC vary widely from those for HTTP ABR streaming.

Broadcast Experience

Required Latency: Reduced (5-18 seconds)

Use Cases: OTT providers; live-streaming news and sports

Interactivity Level: Low

User Experience: One-to-many broadcasts of live events, where streams must compete with cable and satellite TV broadcasts to avoid spoilers for viewers

Best Fit: RTMP; tuned HTTP ABR streaming; cloud-based transcoding

Keep in Mind: For latency near the five-second end of the range, a video switcher mixing multiple cameras with on-screen graphics can be a good fit. Reducing the number of workflow steps (e.g., by combining transcoding and packaging) can help reduce latency even further.

No Time Constraints

Required Latency: Default HTTP (18-45 seconds)

Use Cases: One-way streams of live events to large audiences; linear programming

Interactivity Level: None

User Experience: Live event broadcasts to audiences of 200 simultaneous viewers or more, where interactivity isn't a factor; non-time sensitive programming

Best Fit: Standard HTTP ABR delivery

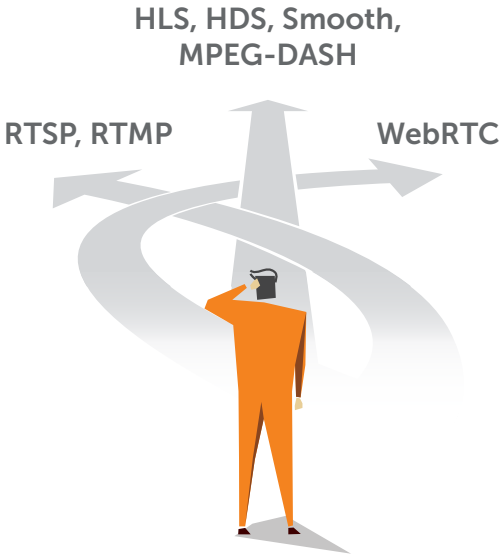
Keep in Mind: Audiences of 200-500 simultaneous viewers can likely be handled with direct delivery from a single media server instance (depending on server hardware and the mix of playback devices, operating systems and screen resolutions). Larger audiences may be best served with a content delivery network (CDN) sourcing from a single media server.

WHICH PROTOCOL IS RIGHT FOR ME?

Today, the streaming protocol you choose depends less on whether it can scale, and more on your use case. This includes the interactivity levels (and subsequent latency rates) you need, and the device-usage patterns of your audience.

With Flash Player now actively blocked in some browsers—and with the rise of plugin-free alternatives for browsers—RTMP usage for delivery will significantly decline over the next five years. Pundits have long decried the death of Flash Player, but for many years, there were no alternatives that provided equal reliability, quality and market ubiquity.

But now, there are two clear front-runners to replace traditional streaming protocols. For normal- and reduced-latency mass distribution using standard CDNs and other Web infrastructure, HLS currently leads the ABR market. For low- and near real-time latency, WebRTC is well-positioned to pick up market share. As support for WebRTC increases, RTMP's decline will likely hasten as older devices, desktops and browsers are retired.



WHAT DOES THE FUTURE HOLD?

We're moving toward a market where industry standards and technology commoditization enable any-screen reach, all across the streaming and interactivity continuum. Given the number of use cases for audio and video, it's no surprise there is not yet a one-size-fits-all streaming protocol. However, it's now feasible to build an integrated, modern video infrastructure that covers the full spectrum of the streaming latency continuum—from the longer latencies common when using default ABR streaming to the latest, real-time options for two-way communications.

Wowza multi-protocol streaming technologies provide you with future-ready deployment options. For low-latency and real-time communications, Wowza Streaming Engine software can help with a range of use cases—from RTMP and RTSP to modern WebRTC streaming. For latencies similar to cable TV and higher, massively scalable HLS and other HTTP formats are available. What's more, using WebSocket and HTTP capabilities, you can host text chat and data streams on the same server used to stream audio and video.

Regardless of use case, Wowza offers the flexibility to balance your quality and interactivity constraints. With Wowza software, you can explore innovative approaches to delivering streams across the latency continuum.

RESOURCES

Low Latency – What Is It and Who Needs It?
www.wowza.com/blog/what-is-low-latency-and-who-needs-it

Achieving Low Latency With Wowza
www.wowza.com/products/capabilities/low-latency

Leverage WebRTC for Low Latency and Optimized Two-Way Communication
www.wowza.com/products/capabilities/webrtc-streaming-software