December 2022

Improving Livestreaming Latency Using Metadata

Andrew Lewis

Stephen Lewis

Peter Lewis

Improving Livestreaming Latency Using Metadata

ABSTRACT

In live media streaming, latency between media capture at a sender and playback at a receiver is optimized to improve user experience. However, media analysis and editing algorithms (like object/sound/speech recognition) that operate on the stream can introduce delays; thus, media may not be transmittable immediately after capture due to delays introduced by processing and potential modification. This disclosure describes techniques of variable latency streaming, where the playback latency relative to the live edge varies during the playback depending on instructions that are generated based on content analysis of the stream. The instructions can be multiplexed into the live stream as timed metadata samples and demultiplexed by the player application at the receiver. The user can set preferences that dictate whether and how the receiver follows the instructions.

KEYWORDS

- Livestreaming
- Streaming latency
- Variable latency streaming
- Live edge
- Remote player

BACKGROUND

In live media streaming, latency between media capture at a sender and playback at a receiver is optimized to improve user experience. For example, optimally, in a low-latency stream of a basketball game, viewers see a goal scored right after it happens; similarly, video callers can have a conversation without pauses after each person’s speech.

However, media analysis and editing algorithms (like object/sound/speech recognition) that operate on a stream can introduce delays. Media may not be transmittable immediately after capture due to delays introduced by processing and potential modifications. For example, a problem in AI-powered, two-way calling using virtual assistants is that audio streaming starts not upon call initiation but only after the identity of the called party is understood by the virtual assistant. Although the intended called party can possibly be detected based on message content, such detection introduces a delay prior to start of audio streaming, which in turn delays the time at which the far-end user finishes listening to a message compared to the live edge at the sender. The sender does not get an immediate response from the far-end user, leading to user dissatisfaction.

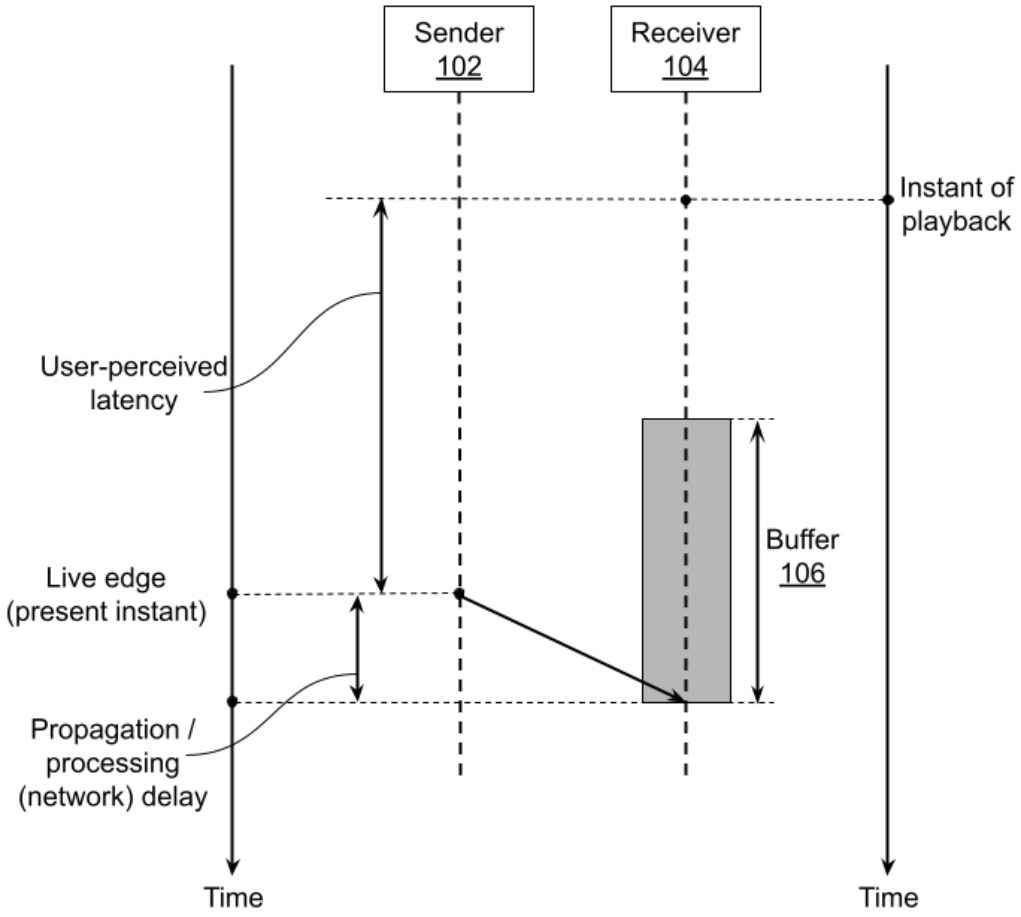

Fig. 1: Example timeline of livestreaming

Fig. 1 illustrates an example timeline of livestreaming. A sender (102) captures video (or audio) at the present instant, also known as the live edge, which is the latest possible playable position in the stream. The sender transmits the captured content to the receiver (104), which receives the content after a propagation and processing delay, also referred to as network delay.

The receiver maintains a buffer (106) to smooth out variations in network delay. The instant of playback is the point in the stream that is currently being played at the receiver. The difference in time between the instant of playback and the live edge is the latency perceived by the user (viewer) at the receiver. The user-perceived latency depends on network delay and on buffer length. As the instant of playback gets closer to the live edge, user-perceived latency is reduced. A low user-perceived latency can enable a satisfying, low-latency video conference (or other content interaction) between sender and receiver. Low-latency can be achieved by reducing the size of the buffer. However, if the buffer is too small, the receiver risks running out of media during periods of high network delays, which can also lead to viewer discomfort.

DASH [3], HLS [4], L-HLS [5], etc. are example media streaming standards with low latency modes. When streaming using these standards, one way to control how close the viewer is to the live edge is to use speed adjustment to go faster or slower than real time. This is used to gradually slow down the received stream if the network latency is going up (to avoid running out of media to play when the stream is arriving too slow) or speed up the received stream if network latency is going down (to get the user closer to the live edge). The player software at the receiver end implements this behavior to adapt to changing network conditions. However, such behavior is independent of the content. Speed changes done gradually are less objectionable than sudden changes, such as having the player pause for an interval, or jump to a different position.

DESCRIPTION

This disclosure describes techniques of variable latency streaming, where the playback (receiver) latency relative to the live edge varies during the playback depending on instructions generated based on content analysis of the stream. These instructions can be multiplexed into the live stream as timed metadata samples, demultiplexed by the player application, and applied at the receiver end. The user can set preferences that dictate whether and how the receiver follows the instructions. The techniques are illustrated through examples.

Example: Livestreaming user-generated content such as presentations

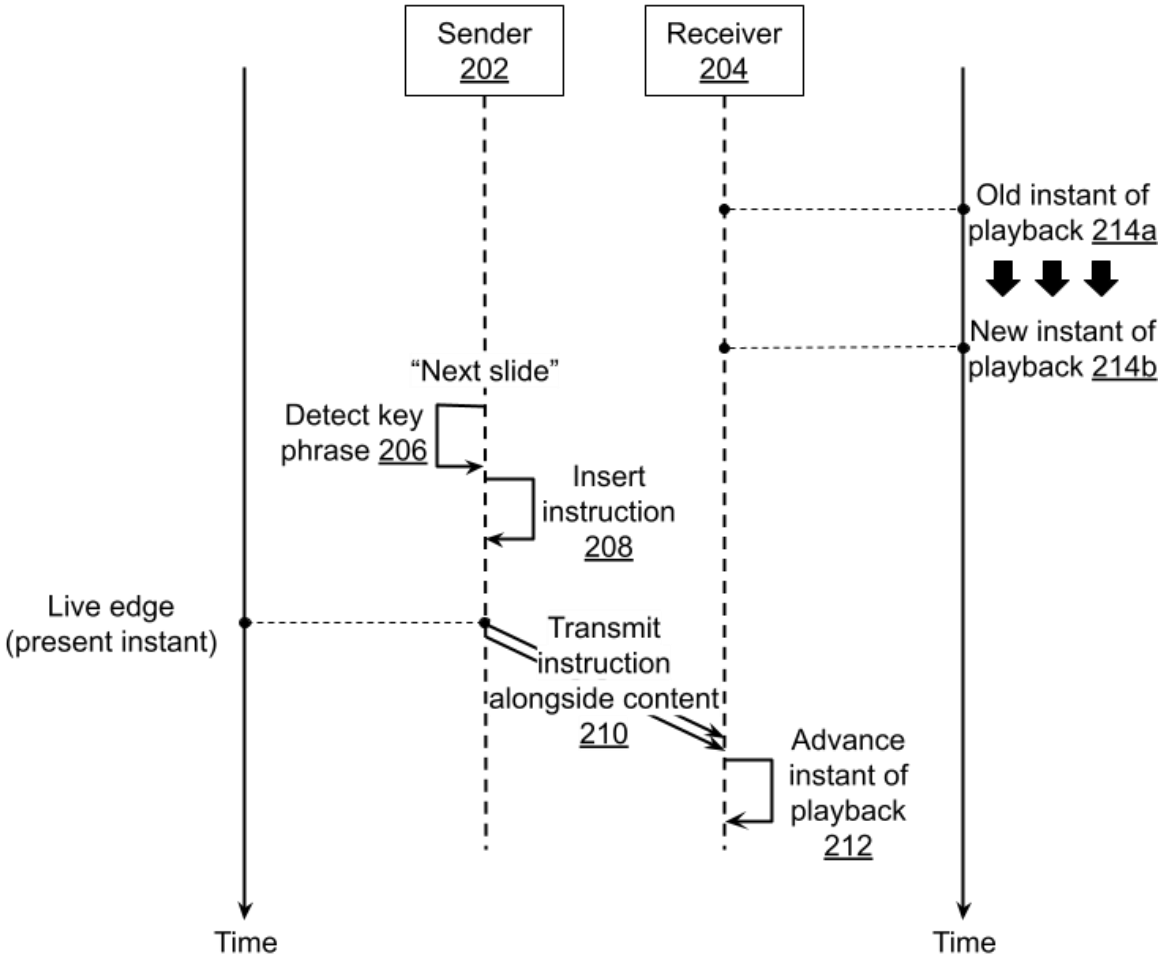

Fig. 2: Improving livestreaming latency in user-generated content using metadata

Fig. 2 illustrates improving livestreaming latency using metadata in user-generated streams, such as presentations. While livestreaming a presentation, a speaker often requests slides to advance by saying “next slide” or another command. At the sending (presenting) end (202), a speech recognizer is applied to detect (206) the utterance and timing of the phrase “next slide.” An instruction is created (208) that includes the time and duration of the phrase and transmitted alongside the content (210). When the livestream is played at the receiver (204), these instructions can be followed to skip the audio where the speaker says “next slide,” enabling the viewer to get slightly closer to the live edge. Skipping the audio can be done by advancing the instant of playback (212) by an amount of time (214a-b) equivalent to the duration of the key phrase “next slide.”

Skipping closer to the live edge on an instance of “next slide,” the player can slow down playback to gradually fall back from the live edge, thereby keeping enough buffer to follow future instructions or to handle network delays without rebuffering.

The described technique of skipping closer to the live edge based on detection of key phrases, audio, or video can be generalized to include filtering portions of livestreamed content in a way customized for the receiver, e.g., removing profanity, out of focus/shaky portions of a video stream, etc. Being computationally expensive, content analysis is optimally done at the sender, while the application of instructions, relying as it does on user preferences, is optimally done at the receiver, though the choice of device to perform such operations can vary and be configured based on the particular use context and/or user preferences. Content analysis that leads to instructions being added to the stream can be based not only on media content but also on other user-permitted factors such as mobile device sensor (e.g., accelerometer) readings, location, audio/video signals, etc.

Example: Video or voice calling using a virtual assistant

As explained earlier, in AI-powered two-way calling using virtual assistants, audio streaming starts not upon call initiation but only after the assistant detects the identity of the person that the sender wants to talk to. Although the intended far-end user can be detected based on message content, such detection introduces a delay prior to start of audio streaming, which in turn delays the time at which the far-end user finishes listening to the message compared to the live edge at the sender. The sender doesn't get an immediate response from the far-end user, leading to dissatisfaction.

Per the techniques described herein, delay is reduced or eliminated at the receiver, also known as remote player, by instructing the remote player to play the message quicker than real-time. To optimize playback speed, e.g., reduce distortion, the instruction can optionally include the deadline at which the message is to finish playing while still hitting the deadline for responsive communication.

Example: Temporarily reducing latency for urgent content in live news or sports events

Some portions of livestreamed content such as news, sports, etc. are more latency sensitive than others. For example, a latency of seconds (relative to the live edge) during advertisements or pre-recorded sections may be acceptable, but a scored goal (or other important event in the sport) or a breaking new story is optimally delivered with negligible latency.

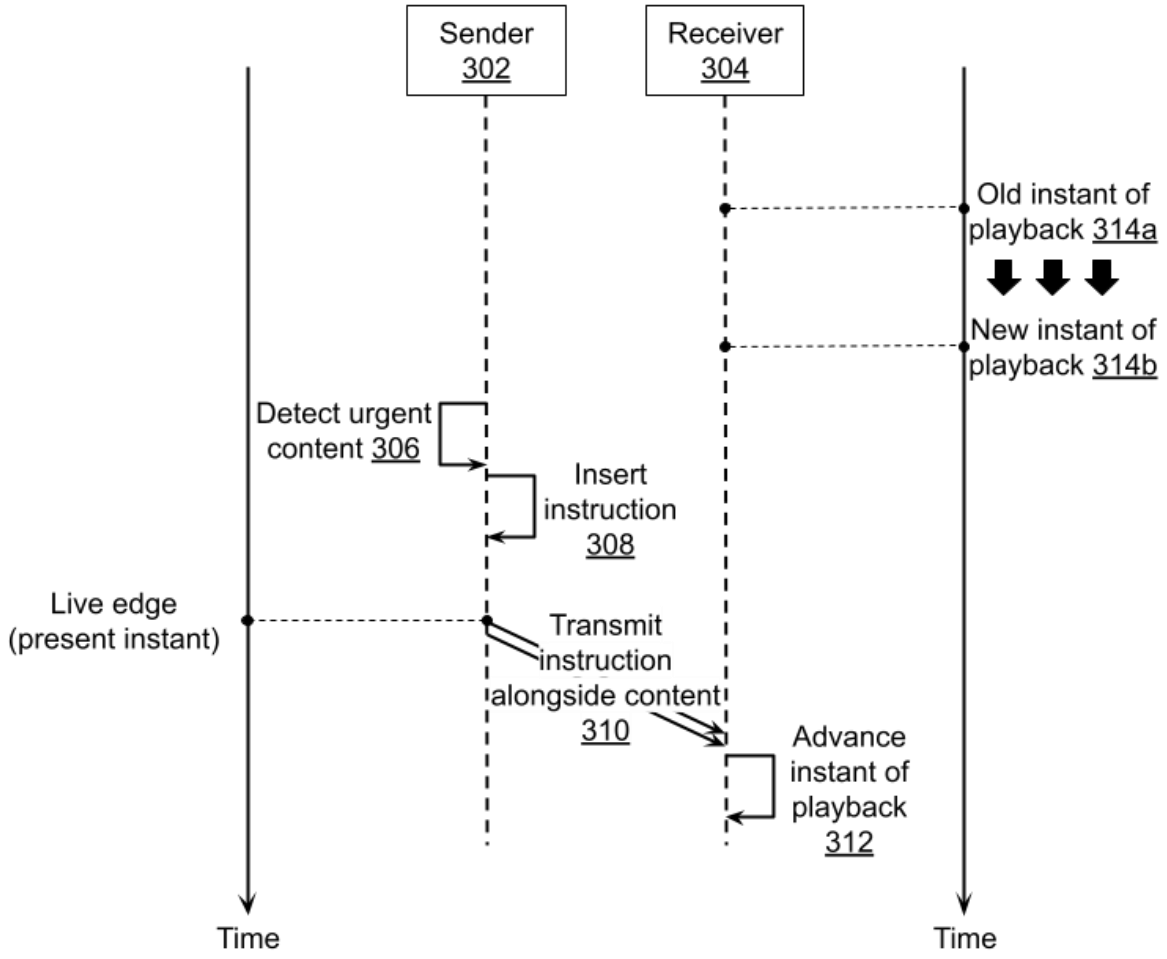

Fig. 3: Improving livestreaming latency in live content (e.g., news or sports) using metadata

As illustrated in Fig. 3, the sender (302) can perform content analysis to detect urgent content (306). Instructions can be inserted into the stream (308) and transmitted alongside the content (310) to guide the receiver (304, also referred to as remote player) to get closer to the live edge for urgent content. The remote player can get closer to the live edge by advancing the instant of playback (312, 314a-b). Conceptually, this is similar to the TCP urgent mechanism. Having a delay relative to the live edge generally enables the receiver to avoid displaying a buffering spinner caused by transient network delays; however, for latency sensitive events, it is worth risking a reduced buffer to ensure that the events are played on time.

In some streaming situations, the remote player can be expected to rewind and replay particular sections of content. Such use cases can be accommodated by having the sender insert synchronization samples (key frames) as well as instructions to enable the remote player to seek backwards and replay quickly without keeping decoded content in memory. If the high likelihood of replay is only known after the start of the replayed section, the sender can selectively re-encode that section before sending the instruction. Receivers can use the instructions as a hint to increase the amount of temporarily cached data.

CONCLUSION

This disclosure describes techniques of variable-latency streaming, where the playback (receiver) latency relative to the live edge varies during the playback depending on instructions generated based on a content analysis of the stream. These instructions can be multiplexed into the live stream as timed metadata samples and demultiplexed by the player application at the receiver. The user can set preferences that dictate whether and how the receiver follows the instructions.

REFERENCES

[1] Potetsianakis, Emmanouil. “Enhancing video applications through timed metadata.” PhD diss., Université Paris-Saclay (ComUE), 2019.

[2] “Embedding Metadata within a Video Stream” available online at https://docs.aws.amazon.com/ivs/latest/userguide/metadata.html accessed Nov. 17, 2022.

[3] “DASH-IF/DVB Report on low-latency live service with DASH” available online at https://dashif.org/docs/Report%20on%20Low%20Latency%20DASH.pdf accessed Nov. 17, 2022.

[4] “Enabling low-latency HTTP live streaming (HLS)” available online at https://developer.apple.com/documentation/http_live_streaming/enabling_low-latency_http_live_streaming_hls accessed Nov. 17, 2022.

[5] “Introducing LHLS media streaming” available online at https://medium.com/@periscopecode/introducing-lhls-media-streaming-eb6212948bef accessed Nov. 17, 2022.

[6] “Audio time stretching and pitch scaling” available online at https://en.wikipedia.org/wiki/Audio_time_stretching_and_pitch_scaling#:~:text=Time%20stretching%20is%20often%20used,as%20a%201%2Dhour%20broadcast. accessed Nov. 17, 2022.

[7] “Semantics of urgent indications” available online at https://datatracker.ietf.org/doc/html/rfc6093#section-2.1 accessed Nov. 17, 2022.