Timestamp Troubles

How Mux handles unreliable system clocks in virtual environments

Walker Griggs

October 13, 2022

whoami

Currently at **Mux** on Live Studio team

Previously at **Heroku** on Data Team

No reason for being this bad at Chess, for the amount that I play

walker@mux.com - walkergriggs.com

MUX

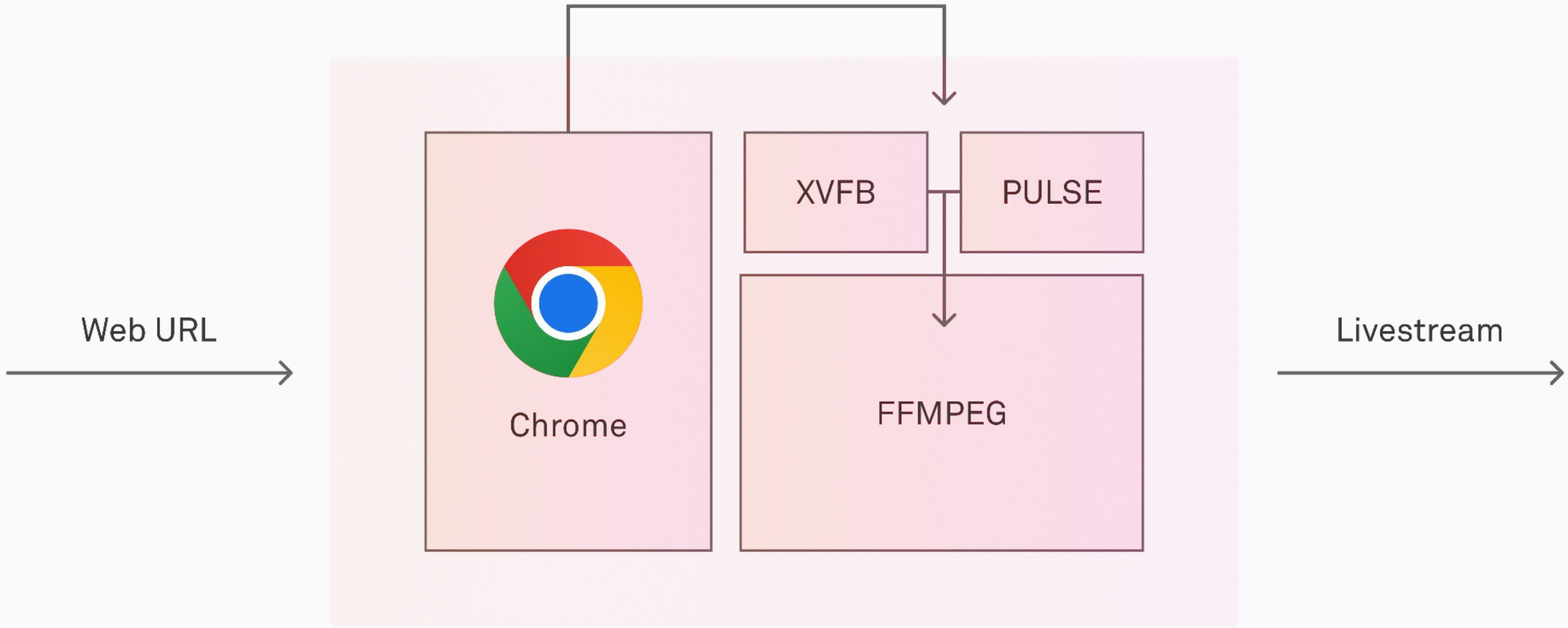
Reliable timestamps when live streaming from virtual environments are really hard

Agenda

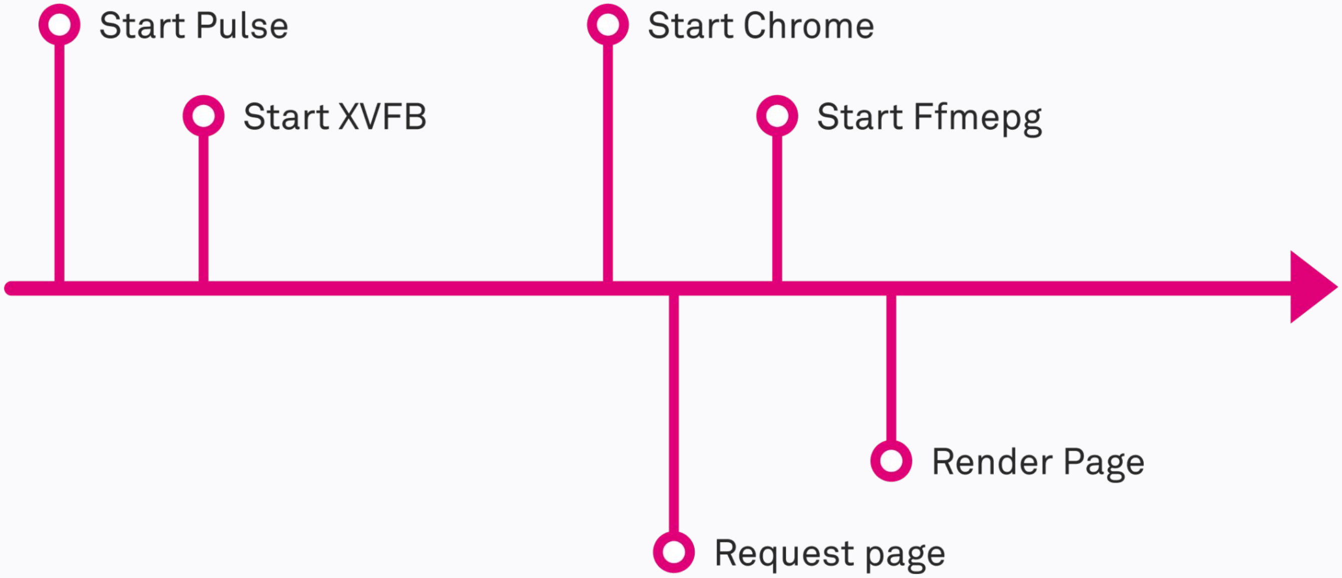
1. Web Inputs and unexpected behaviors
2. A bit about timestamps
3. Our triage journey
4. Fixing it!

1. **Web Inputs and unexpected behaviors**
2. A bit about timestamps
3. Our triage journey
4. Fixing it!

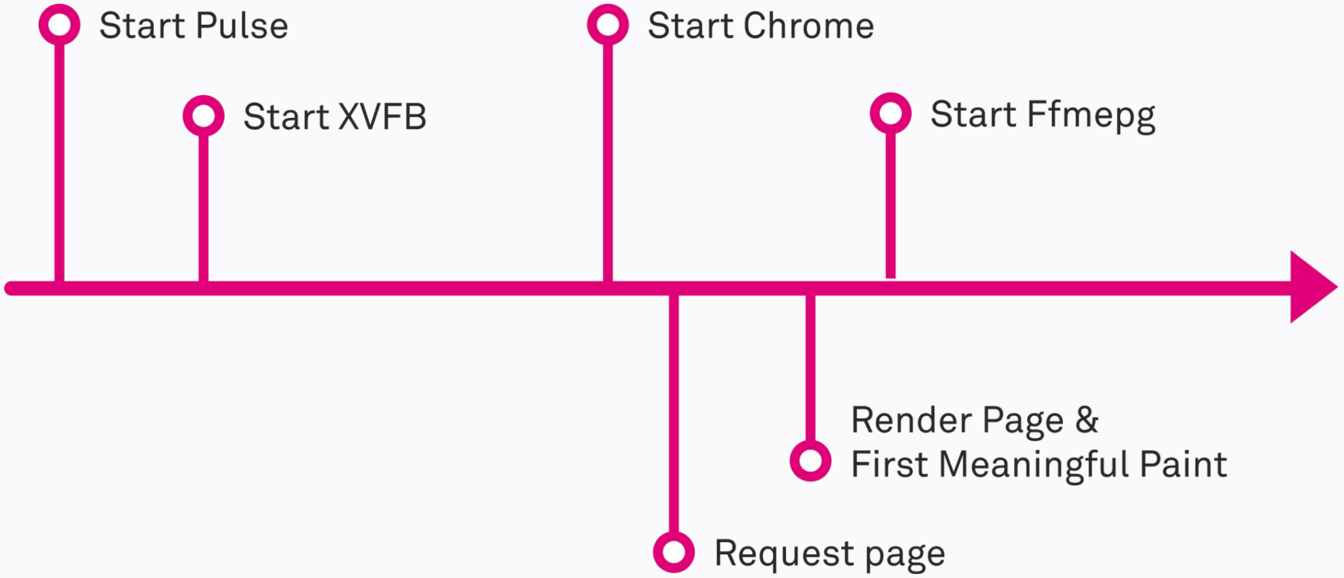
Web Inputs at a glance



Process timeline

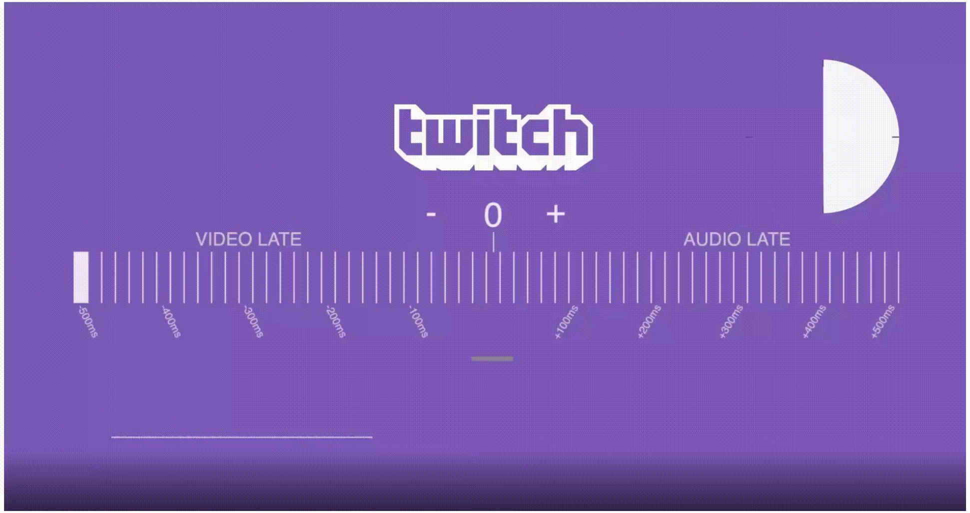


Process timeline



Behavior #1:
Scattered audio and video for the first few seconds

Behavior #2:
Audio and video sync meander throughout the livestream



Easily fix async video with ffmpeg

Feb 17, 2021 • Lars Windolf –

1. Correcting Audio that is too slow/fast

This can be done using the `-async` parameter of ffmpeg which according to the documentation *"Stretches/squeezes" the audio stream to match the timestamps*. The parameter takes a numeric value for the samples per seconds to enforce.

```
ffmpeg -async 25 -i input.mpg <encoding options> -r 25
```

Try slowly increasing the -async value until audio and video matches.

2. Auto-Correcting Time-Shift

2.1 Audio is ahead

ffmpeg(1) - Linux man page

Name

ffmpeg - FFmpeg video converter

Synopsis

ffmpeg [[infile options][**-i** *infile*]]... {[outfile options] *outfile*}...

Description

As a general rule, options are applied to the next specified file. Therefore, order is important, and you can have the same option on the command line multiple times. Each occurrence is then applied to the next input or output file.

* To set the video bitrate of the output file to 64kbit/s:

```
ffmpeg -i input.avi -b 64k output.avi
```

* To force the frame rate of the output file to 24 fps:

```
ffmpeg -i input.avi -r 24 output.avi
```

projects / ffmpeg.git / blobdiff

summary | shortlog | log | commit | commitdiff | tree
raw | inline | side by side

avdevice/pulse_audio_dec: do not read undersized frames

[ffmpeg.git] / libavdevice / pulse_audio_dec.c

```
diff --git a/libavdevice/pulse_audio_dec.c b/libavdevice/pulse_audio_de
index 0454a643dda8bd2b7170db5b0288392f87b95337..3777396ef60d2e30922d52
--- a/libavdevice/pulse_audio_dec.c
+++ b/libavdevice/pulse_audio_dec.c
@@ -48,6 +48,7 @@ typedef struct PulseData {
     pa_threaded_mainloop *mainloop;
     pa_context *context;
```

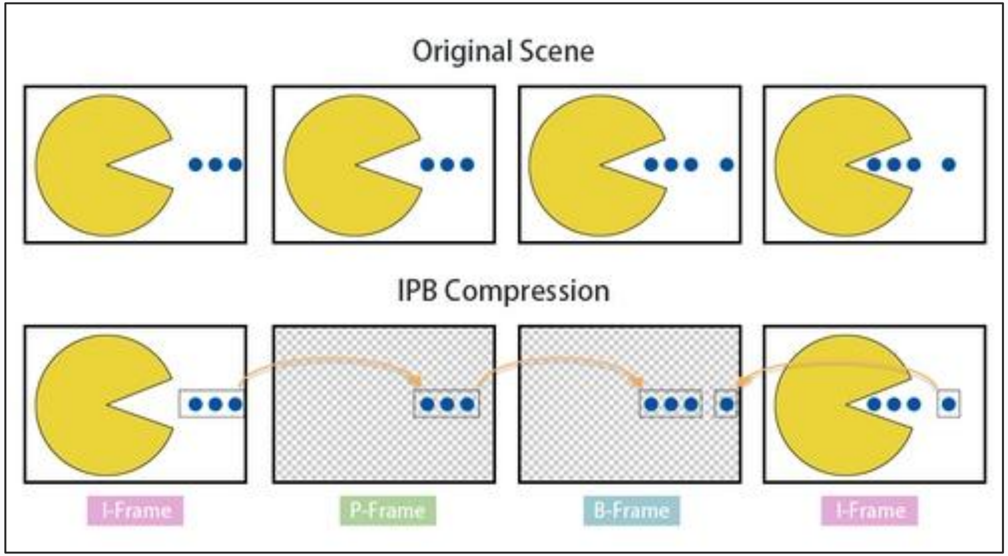
1. Web Inputs and unexpected behaviors
2. **A bit about timestamps**
3. Our triage journey
4. Fixing it!

PTS, Presentation Timestamps
When the player should show you the frame

DTS, Decode Timestamps
When the player should decode the frame.

Some frames are predictive and reference other frames.

Predictive frames change the order they should be decoded in.



1. Web Inputs and unexpected behaviors
2. A bit about timestamps
3. **Our triage journey**
4. Fixing it!

Non-monotonic DTS in output
Decoded timestamps are out of order!

Buffered samples
Packet sizes are initially 64kb, but settle to 4kb

DTS and PTS in audio samples?
Structs are generic for both audio and video. Audio samples aren’t ‘predictive’

[flv @ 0x47b5740]
Non-monotonous DTS in output stream 0:1; previous: 320, current: 3; changing to 320. This may result in incorrect timestamps in the output file.

[pulse @ 0x47ac840]
DTS: Unix Timestamp,
PTS: Unix Timestamp + 3410000,
Latency: 210971,
Frame Duration: 16368,
Read Length: 65472,
Frame Size:4

```
// Grab the wallclock
dts = av_gettime();

// Adjust for latency
dts +/-= pa_stream_get_latency(...);

// Denoise the adjusted timestamp
pts = ff_timefilter_update(...);
```

Non-monotonic DTS in output

Decoded timestamps are out of order!

Buffered samples

Packet sizes are initially 16kb, but settle to 4kb

DTS and PTS in audio samples?

Structs are generic for both audio and video. Audio samples aren’t ‘predictive’

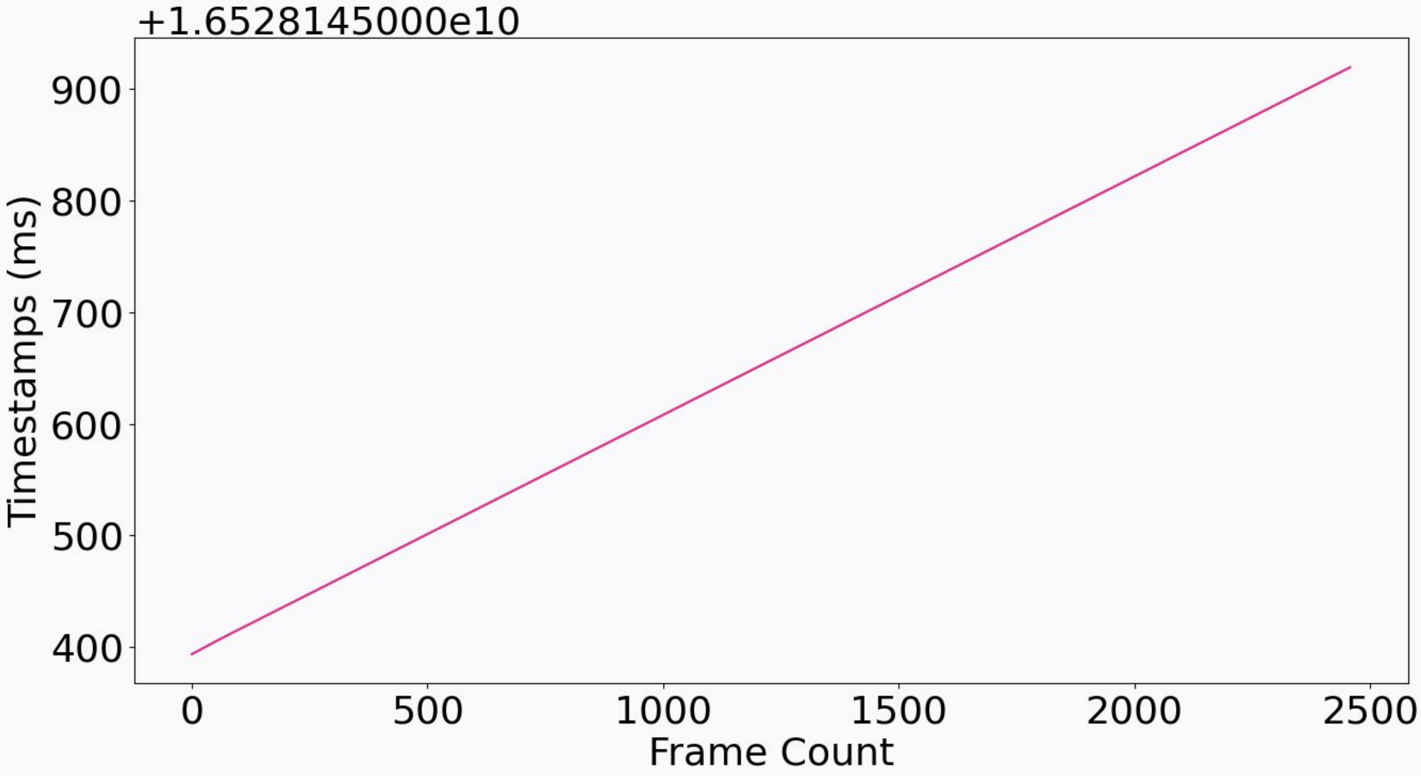
[flv @ 0x47b5740]

Non-monotonous DTS in output stream 0:1; previous: 320, current: 3; changing to 320. This may result in incorrect timestamps in the output file.

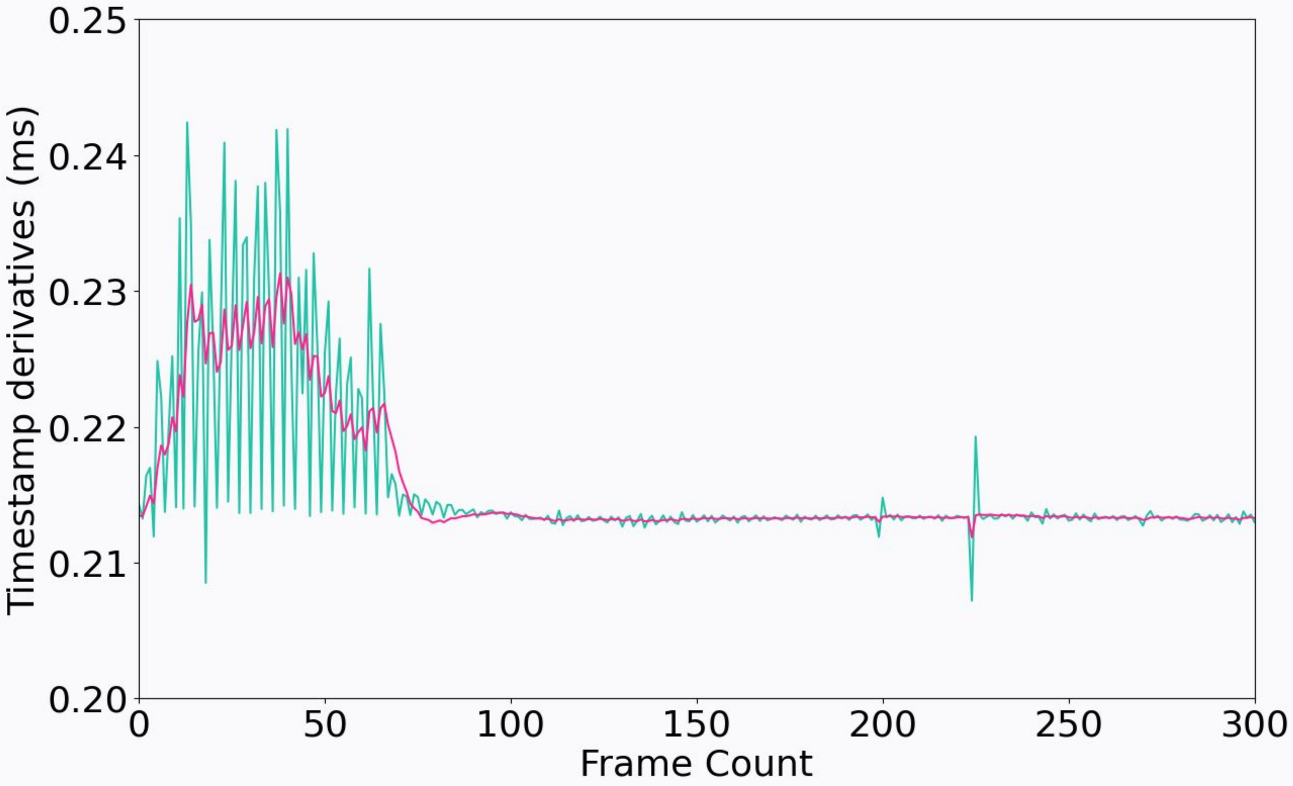
[pulse @ 0x47ac840]

DTS: Unix Timestamp,
PTS: Unix Timestamp + 341000,
Latency: 210971,
Frame Duration: 16368,
Read Length: 65472,
Frame Size:4

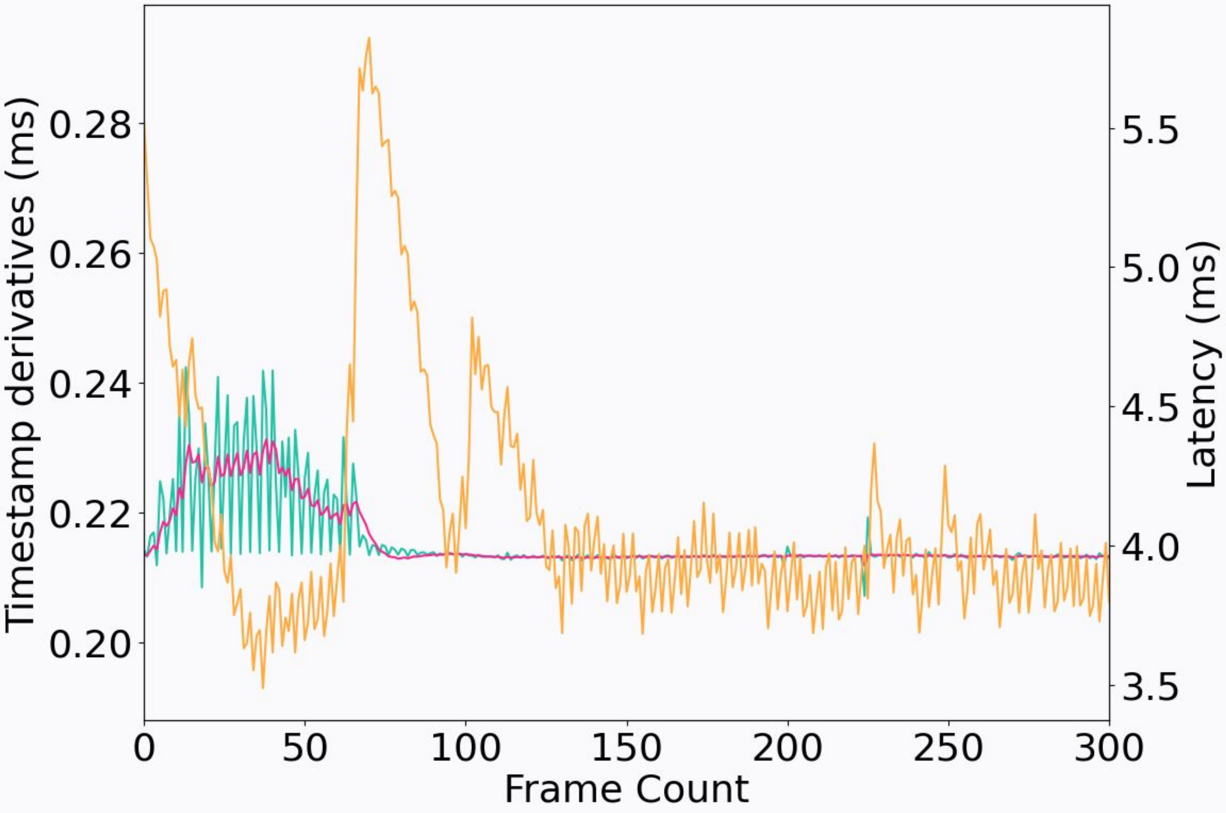
DTS over Frame Count



DTS Derivative over Frame Count

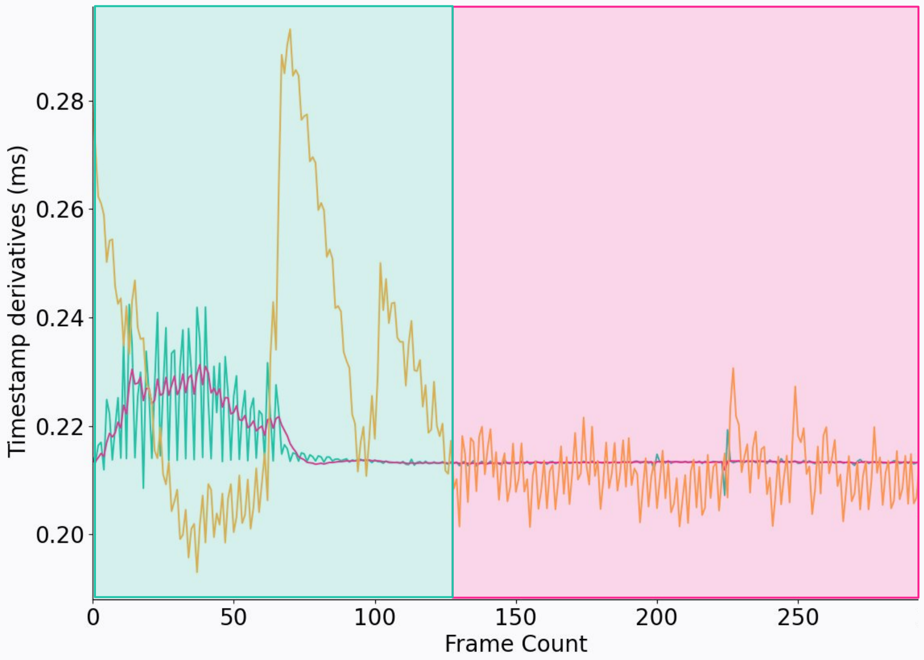


DTS Derivative over Frame Count

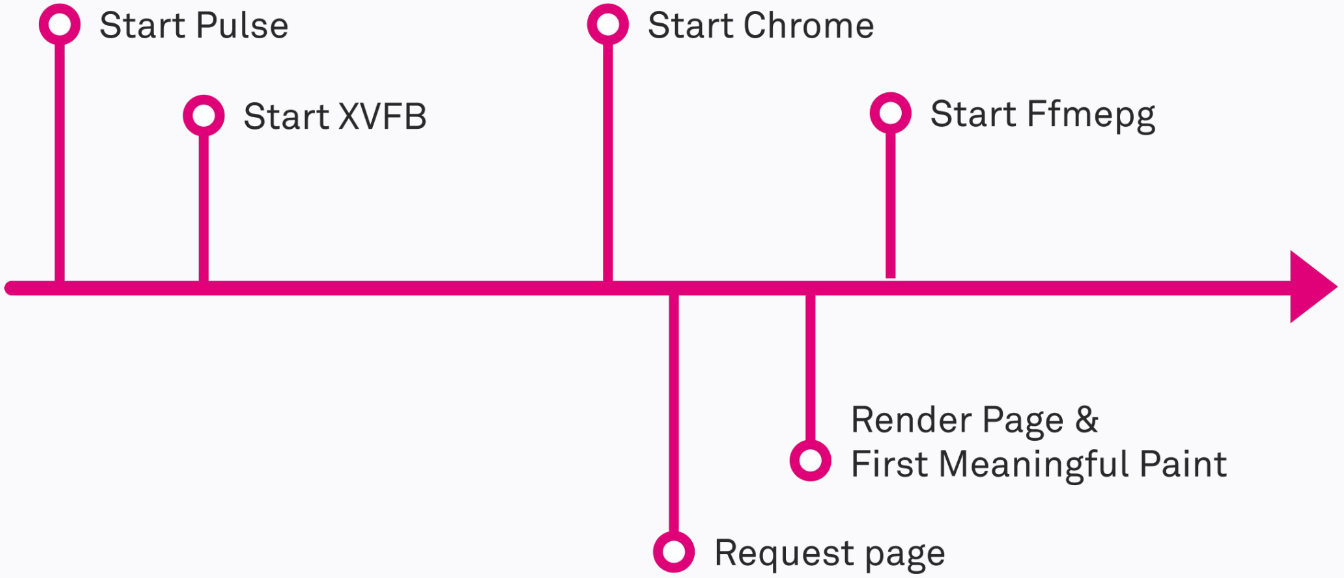


Behavior #1:
Scattered audio and video for the first few seconds

Behavior #2:
Audio and video sync would meander throughout the livestream



Process timeline



1. Web Inputs and unexpected behaviors
2. A bit about timestamps
3. Our triage journey
4. **Fixing it!**

What do we know?

Pulse is buffering more than we need
We don’t need to transcode those samples in the first place.

Wall clock isn’t perfect
Even after de-noising, it still fluctuates

We know the useful decoding metrics
Target frequency, total number of samples decoded, and starting timestamp

What did we do?

Ignore large packets
Only decode nice, round 4kb packets. Naive!

Flush the Pulse buffers
Call the Pulse API to flush the buffers directly from device decoder.

Count samples
Computing a timestamp is as simple as Starting Time + (Samples / Frequency)

How did we do it?

1. Record the starting timestamp

2. Count the number of samples decoded

3. Ignore samples with an DTS before that starting timestamp

4. Use the target frequency and number of samples to find our PTS

   (Total Samples / Target Frequency) + Starting timestamp

```
pts = init_pts + av_rescale(
                   total_samples,
                   timebase,
                   sample_rate
                 );
```

What gives?

Counting samples isn't responsive
This system won't recover if the sync is off.

Sharks bite cables
There are a number of reasons why we might lose audio samples. Entropy exists

System of checks and balances
Use the wall-clock to check if we've drifted by more than some threshold

Are you learning, son?

Get your hands dirty!
Fill the gap between glossary and technical specification.

Choose redundancy where it matters
You can’t trust any single systems.

Invest in glass-to-glass testing, early
If I have to listen to one more test card...

“Reliable timestamps when live streaming from virtual environments are really hard”