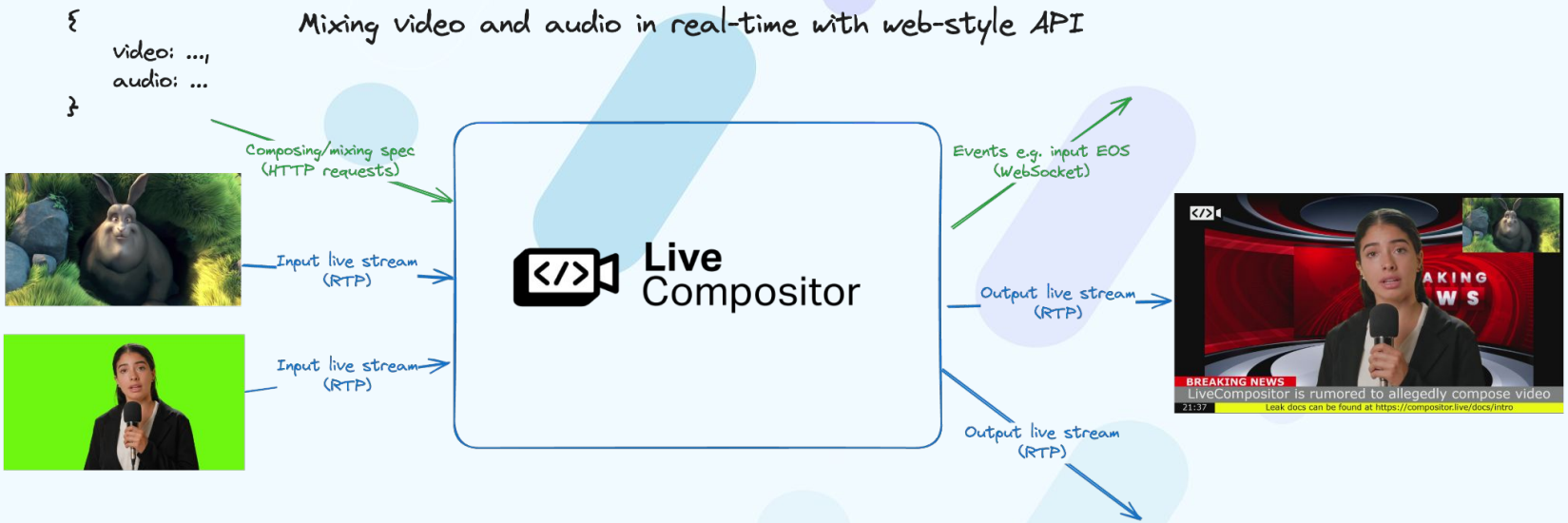
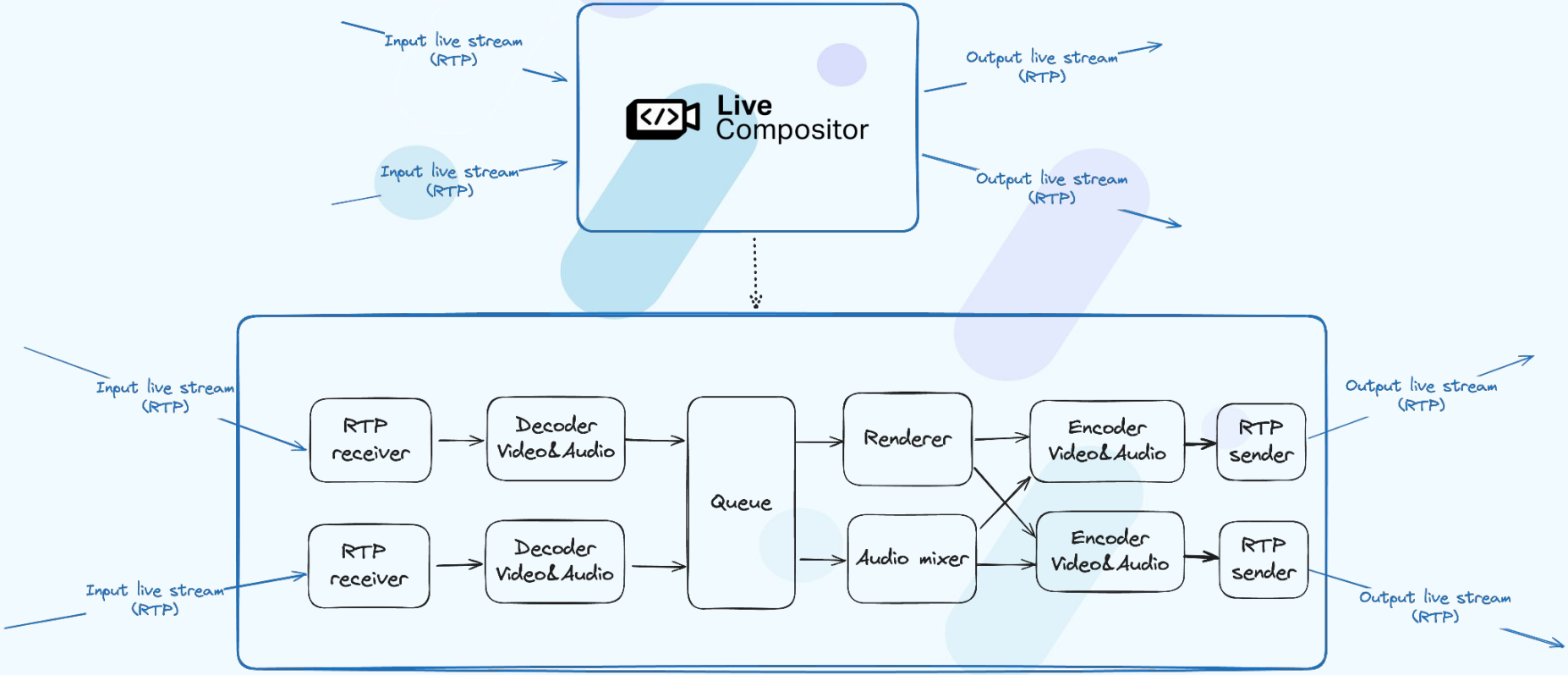
Creating React for live streams
Low-latency multimedia insights

Wojciech Barczyński

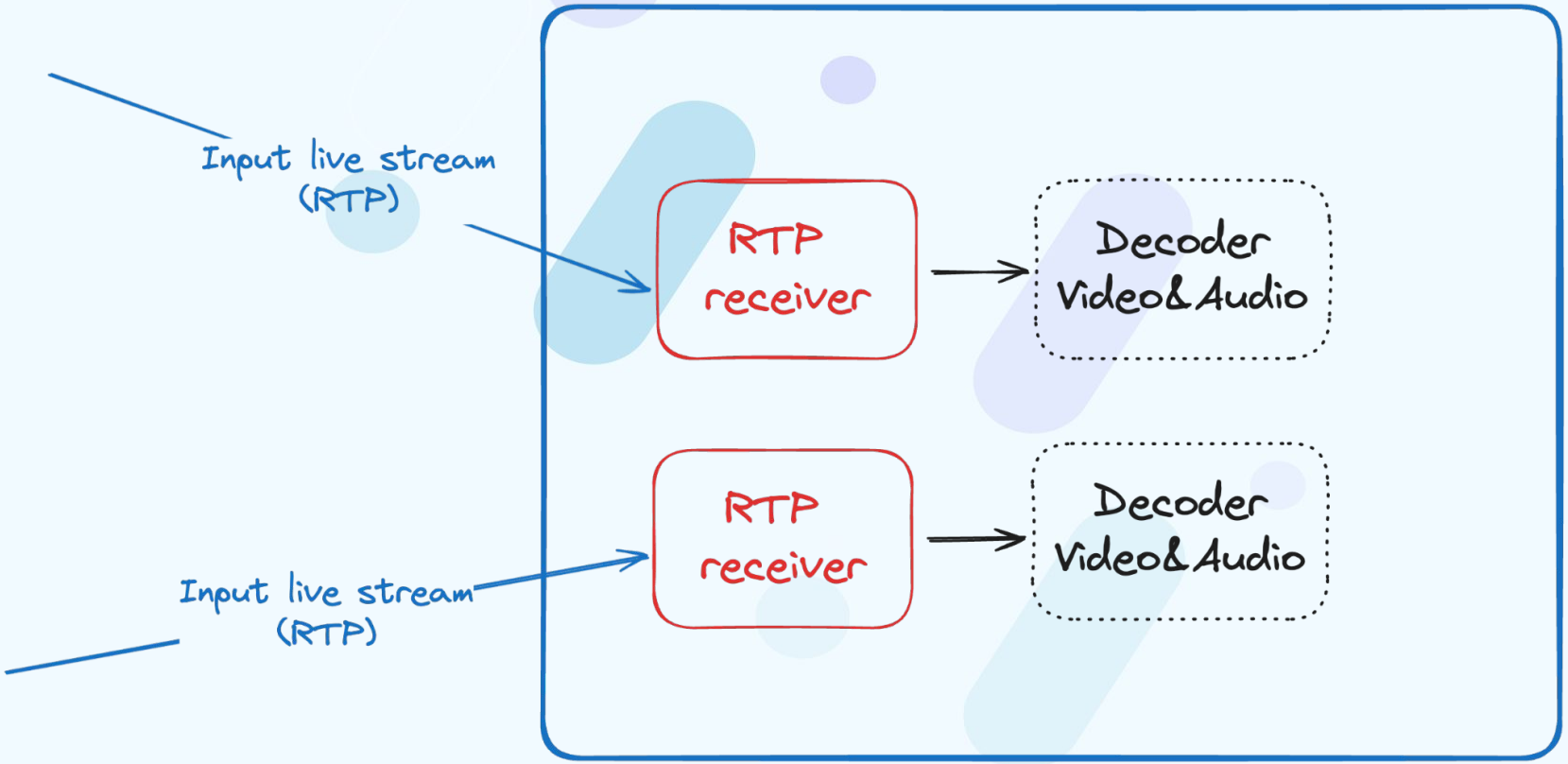
React for live streams?



How it works?


Input live stream (RTP)
Input live stream (RTP)
Live Compositor
Output live stream (RTP)
Output live stream (RTP)
Input live stream (RTP)
Input live stream (RTP)
RTP receiver
RTP receiver
Decoder Video&Audio
Decoder Video&Audio
Queue
Renderer
Audio mixer
Encoder Video&Audio
Encoder Video&Audio
RTP sender
RTP sender
Output live stream (RTP)
Output live stream (RTP)

Let's get started!



Weird 50% packet loss… on localhost?

Observed issues

- Weird packet loss on localhost while streaming with RTP
- A few first packets were correct
- Many next packets were lost
- Only happened on Linux - worked fine on MacOS

UDP buffer overflow

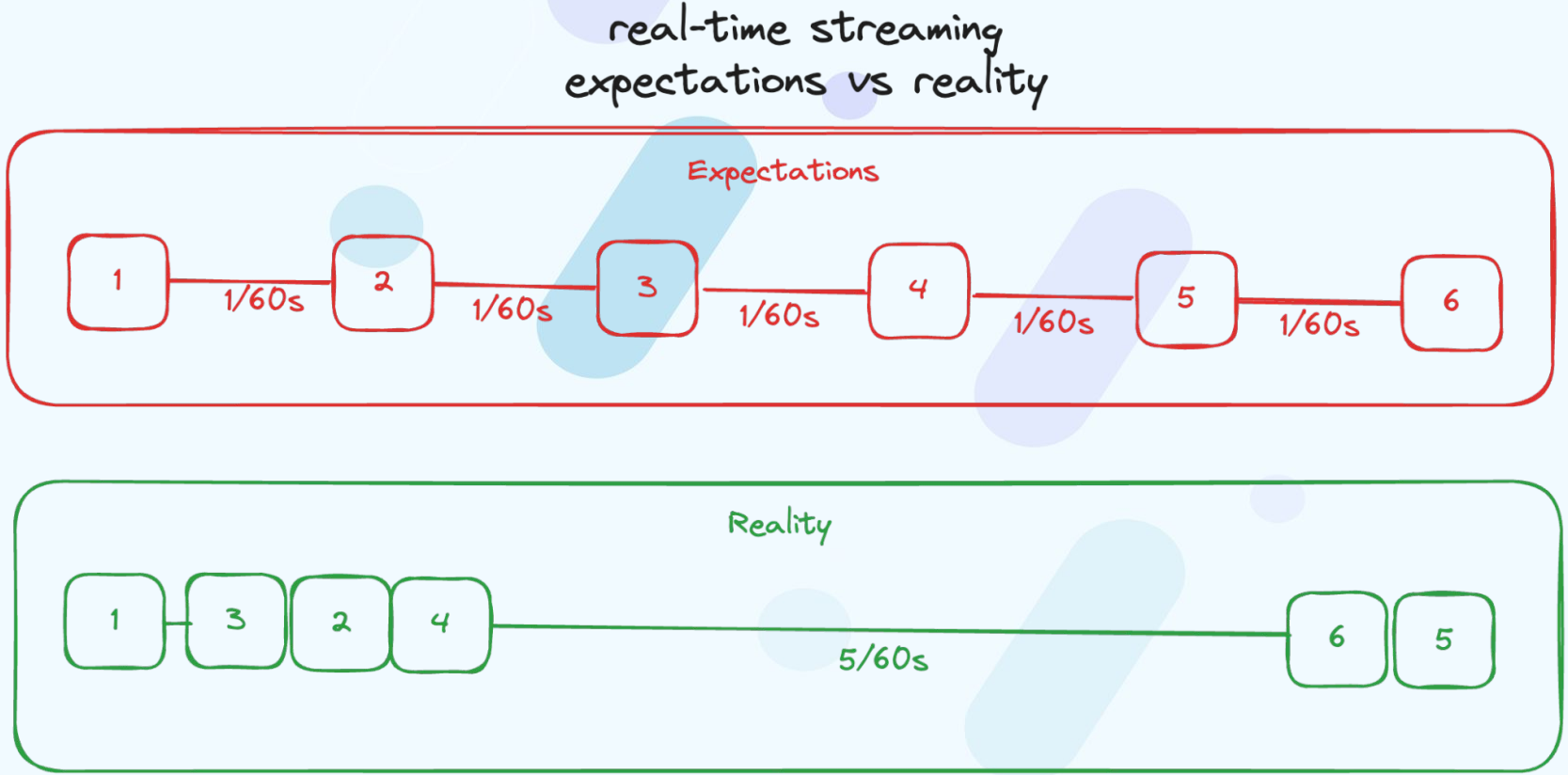
Problems:

- **By default UDP buffers on Linux are small (usually ~200kB)**
- **UDP packets are discarded and lost if overflow occurs**

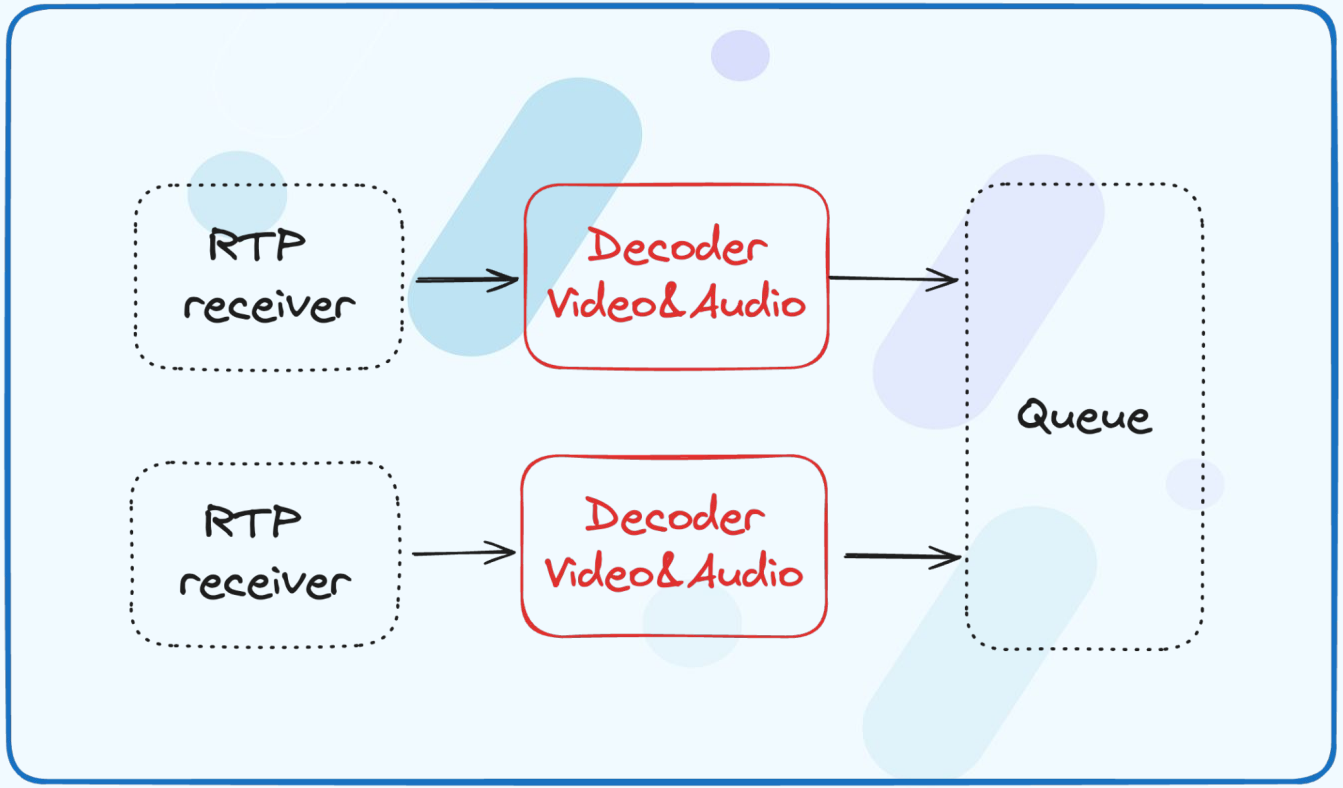
Solutions:

- **Increasing UDP buffer size**
- **Speeding-up reading from UDP sockets**

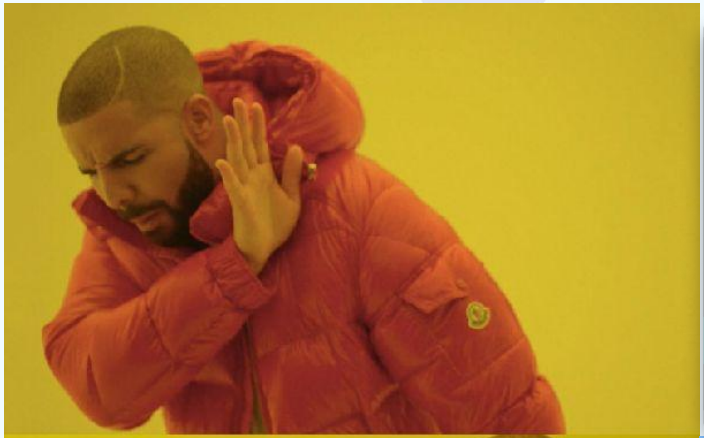
Usually packets are coming in batches

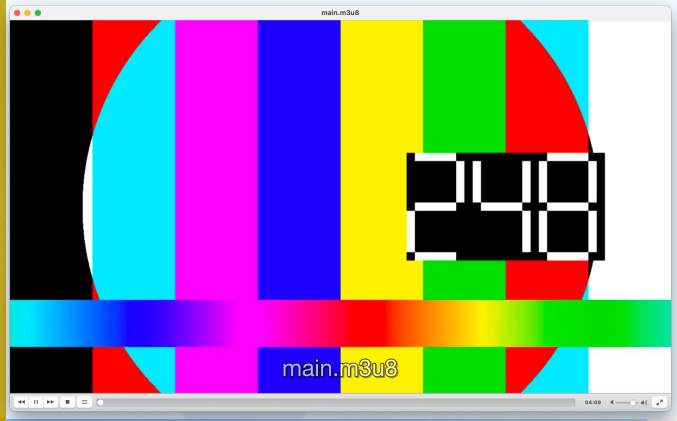


Decoders


RTP receiver
Decoder Video&Audio
RTP receiver
Decoder Video&Audio
Queue

Test performance with “real” video

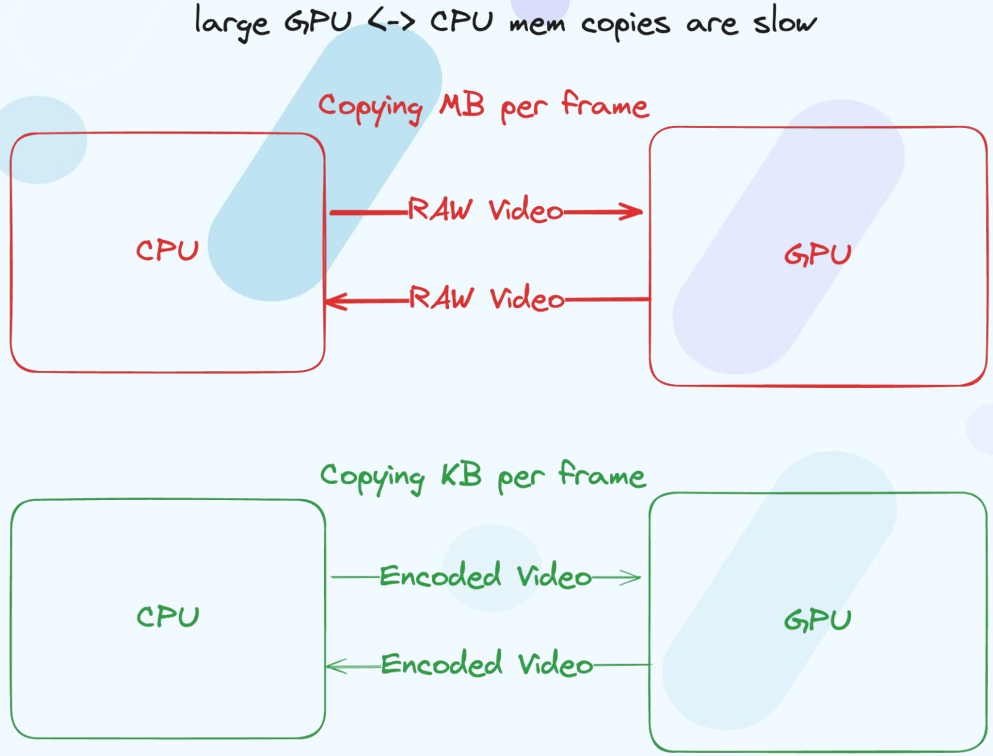




Hardware decoding/encoding

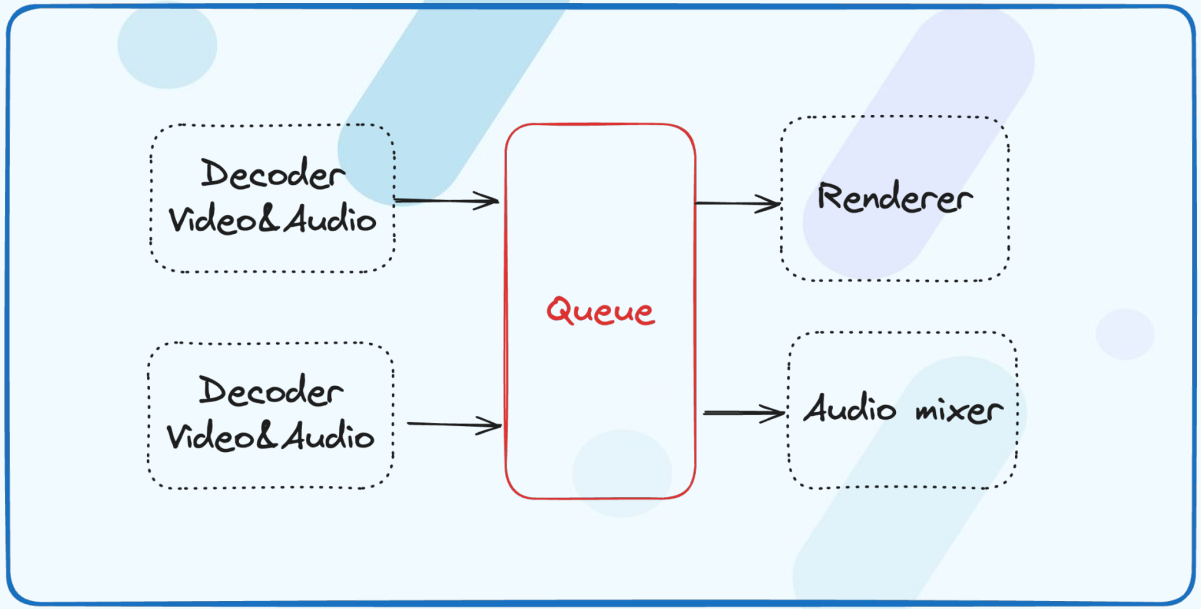
- **Decoding/encoding is often the most computationally expensive part of the pipeline**
- **With FFmpeg / GStreamer changing decoders/encoders is quite easy**
- **Low-hanging fruit in optimization process**

Reducing memory copying with hardware decoding/encoding



Queue – one element to rule them all!

Synchronizing streams, lazy decoding, network instability handling



Queue

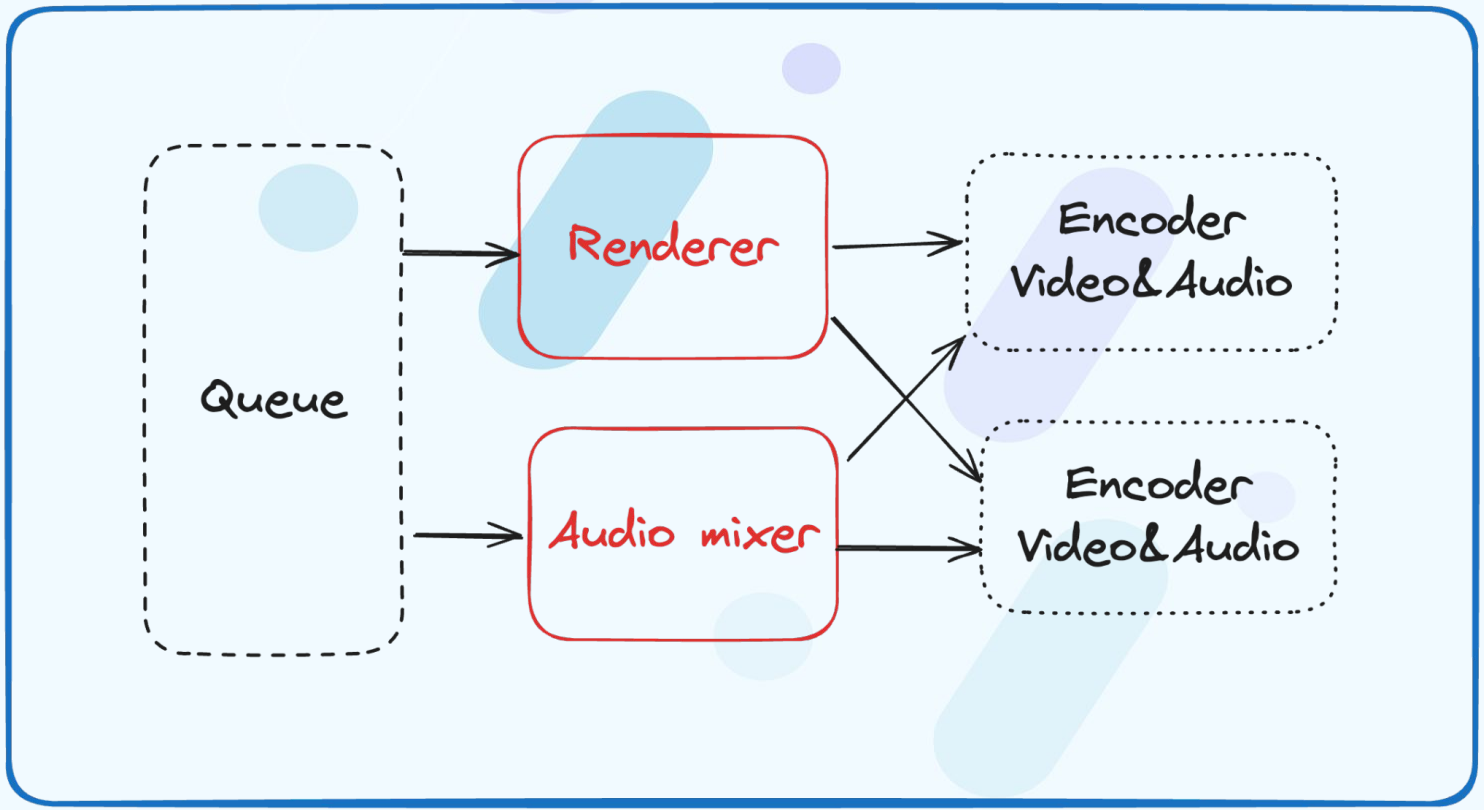
Buffering:

- **Reducing impact of network unreliability**
- **Helps stream synchronization**

Back-pressure:

- **Reducing RAM usage with lazy decoding**

Rendering / Mixing



Preallocating memory

- Memory allocations are slow
- It’s faster to copy memory to preallocated buffer, than allocating memory “on the fly”

What we did?

- GPU textures are preallocated - we don’t need to do large allocations on each render
- LiveCompositor sometimes use stack preallocated array buffers instead of heap OTF allocated data structures

Profiling – check your hardware

With GPU rendering:

- **Decoding & encoding - >70% CPU usage**
- **Rendering - <1% CPU usage**
- **Rendering cost is negligible**

With emulating rendering on CPU:

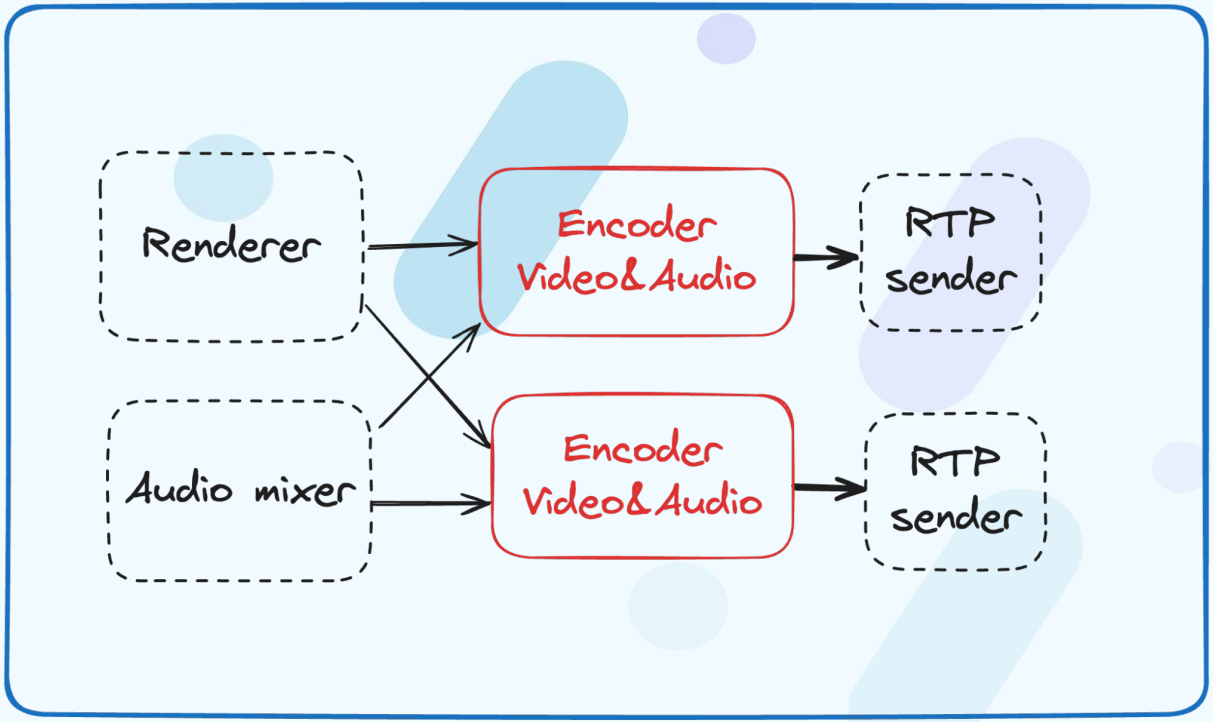
- **Rendering - ~70% CPU usage**
  (depends on render complexity)

Profiling – check your hardware

We optimized renders to be able to run on servers without GPU by:

- **Rendering text/image textures only once**
- **Flattening all “div”-s to render them at once**

Encoders – low effort optimizations



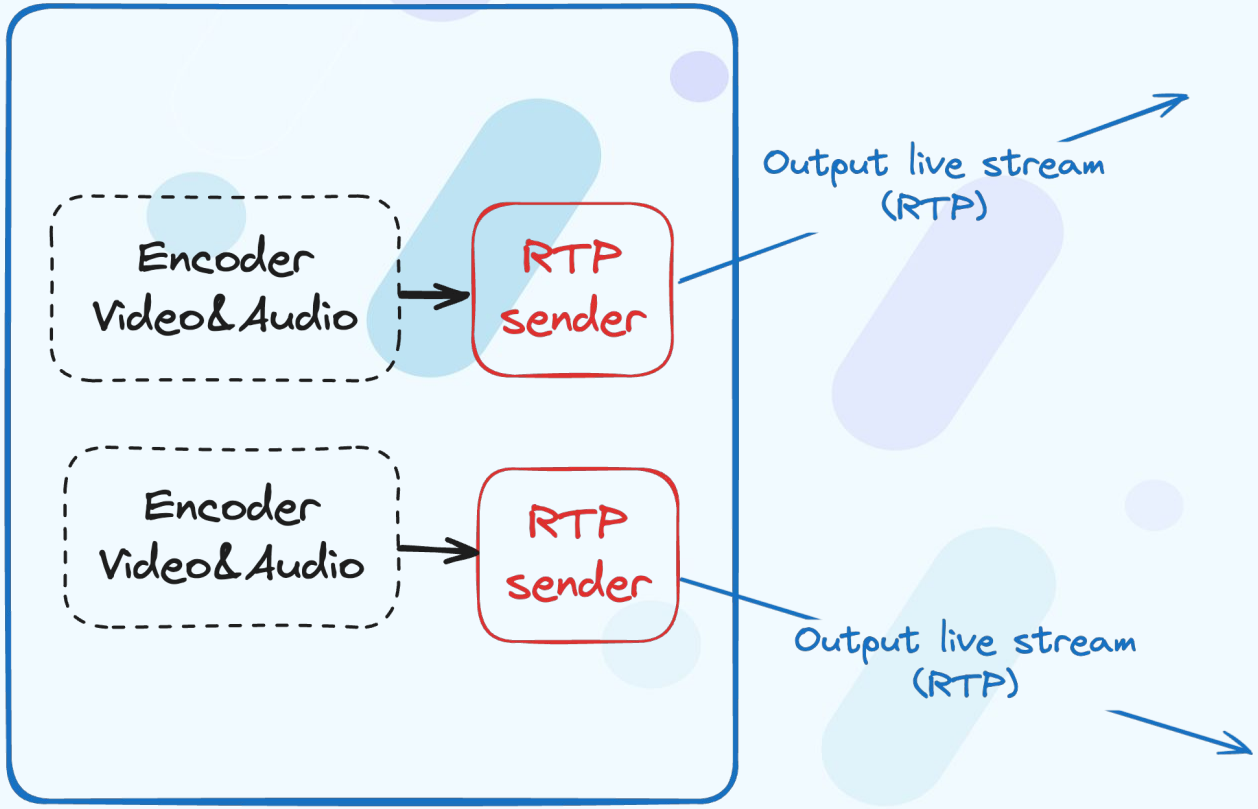
Encoders – low effort optimizations

Hardware encoding has worst quality, but is much faster.

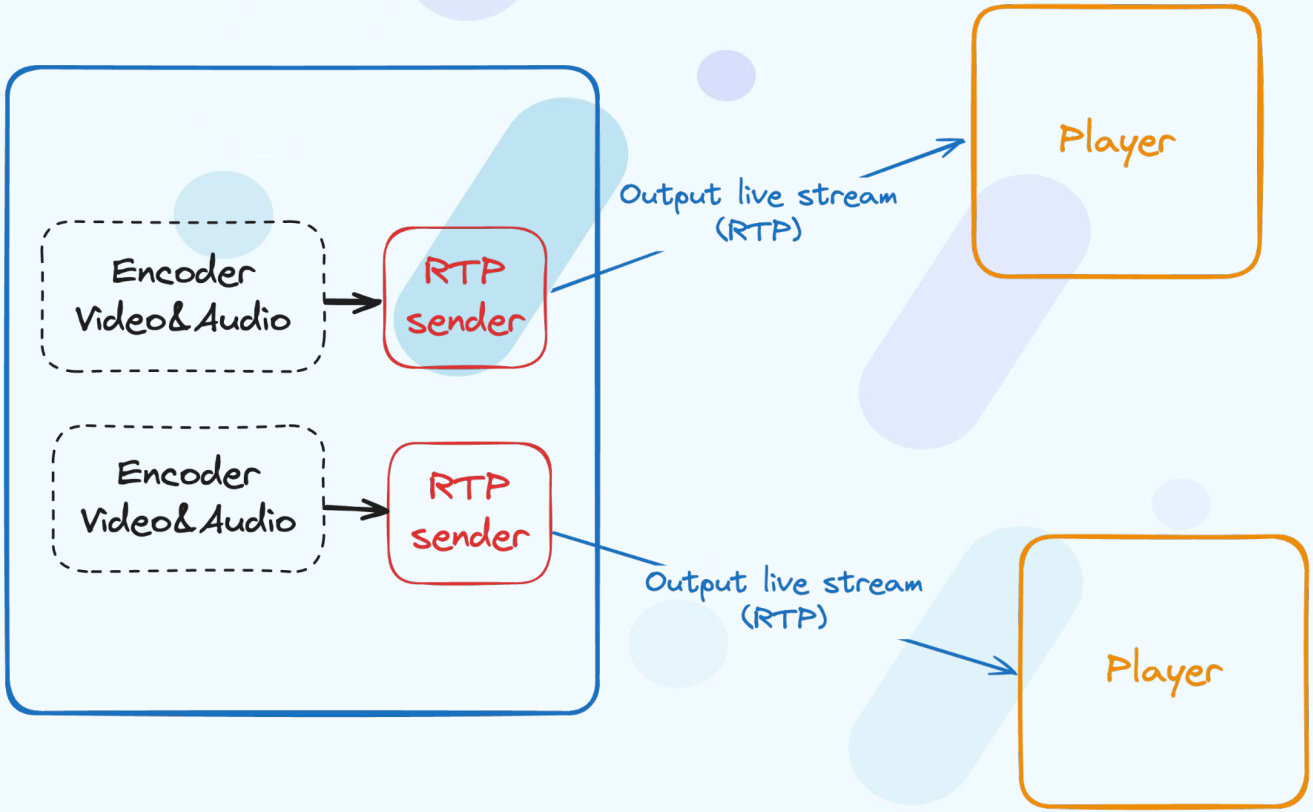
Changing encoder setting can greatly impact your performance:

- **Encoder preset and bitrate are most important**
- **ip-factor (key-frames / p-frames), tune etc. also have some impact on performance and latency**

Sending outputs



Sending outputs – know your players!



Sending outputs – know your players!

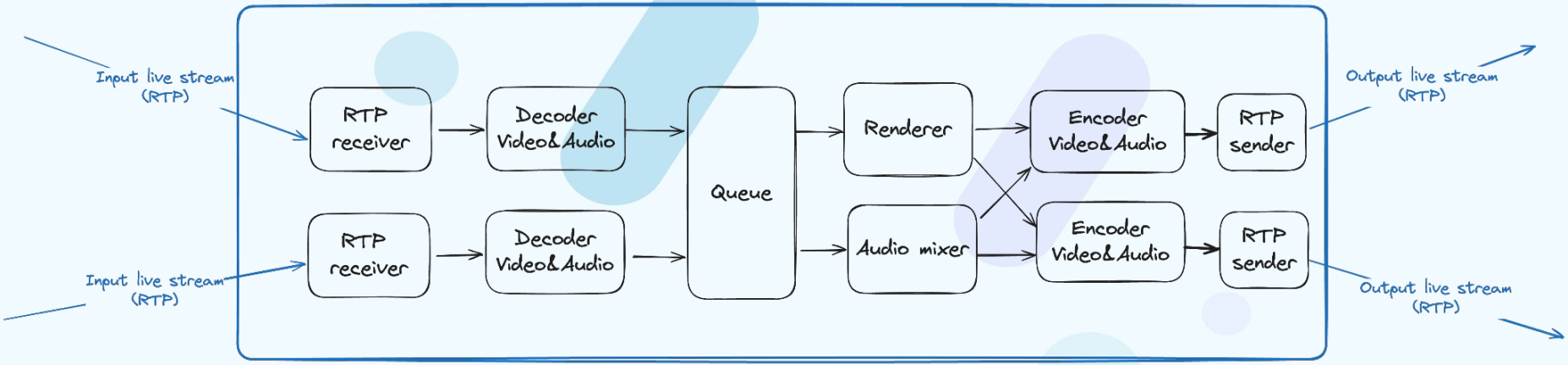
Not all players work in the same way!

Some players can handle metadata differently.

Examples:

- **Different SDP handling**
- **Disregarding audio PTS values**

Live Compositor



Ask important questions

Before optimizing your pipeline, you should consider:

- **Does it have to be real-time?**
- **Costs of the other parts of infrastructure**

Does it have to be real-time?

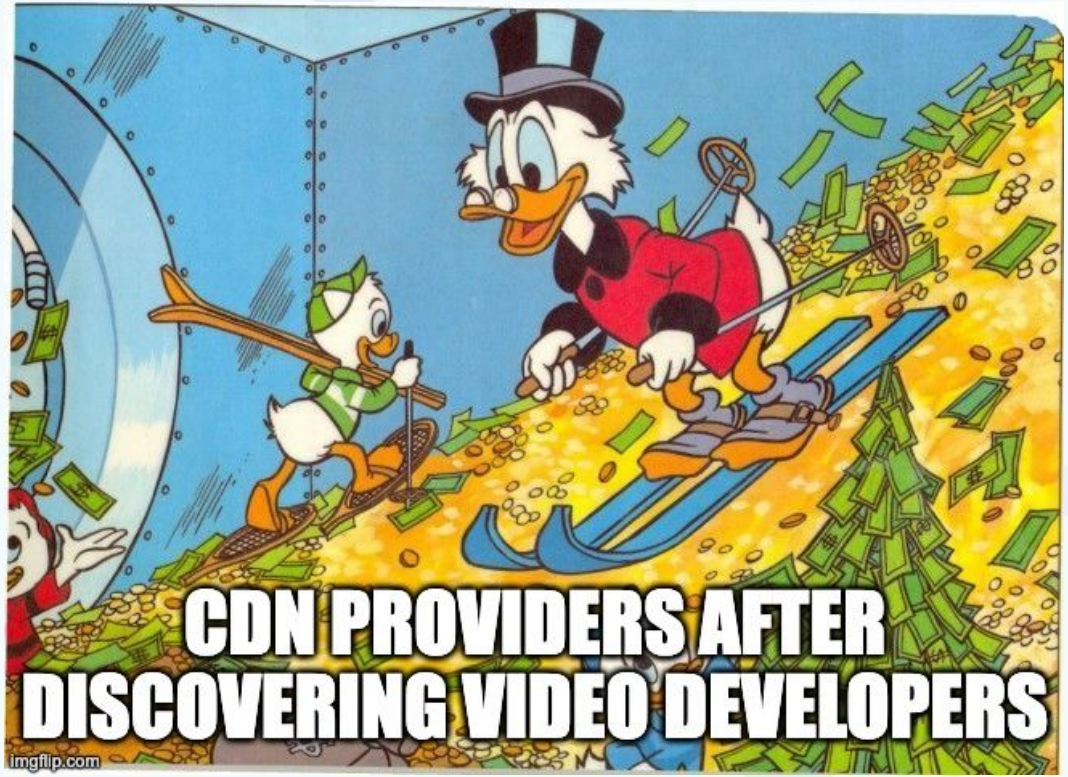
If you can process it offline, maybe it's not worth to do it real-time.

LiveCompositor also supports offline processing

With offline processing you can:

- **reduce complexity**
- **send streams reliably, without losing frames (e.g. TCP vs UDP)**
- **process all streams without dropping frames**
- **perform operations with frame-perfect precision**

Cost of the other parts of infrastructure



Live demo

Announcements

Stable API

11 - 13 SEPT, KRAKÓW, POLAND

RTC.ON

RTC developement conference

RTC.ON newsletter – what do you get

- **Dev tips and tricks** on Membrane, WebRTC, Streaming & Computer vision,
- **Materials from all around the web** that our team found useful and thought you might as well!
- **RTC.ON news** – be the first one to know about ticket sale, lineup, and more!
- **Monthly summary** of everything that happened in Membrane – new releases and exciting news, all in one place,

Join via the QR code and get additional 15% off all tickets!

SCAN & SUBSCRIBE

Questions?

Thanks for listening