IT'S *TIME* FOR SOME  RATE MATCHING!

Arun Raghavan

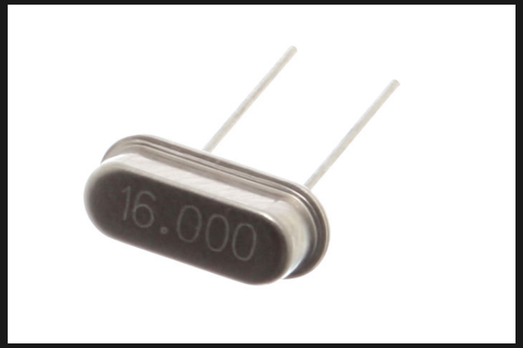
asymptotic

ME

- GStreamer, PipeWire, PulseAudio person
- Founder @ asymptotic
  - Help build cool things with these projects
  - Contribute back to upstream better
  - ???
  - Positive feedback loop

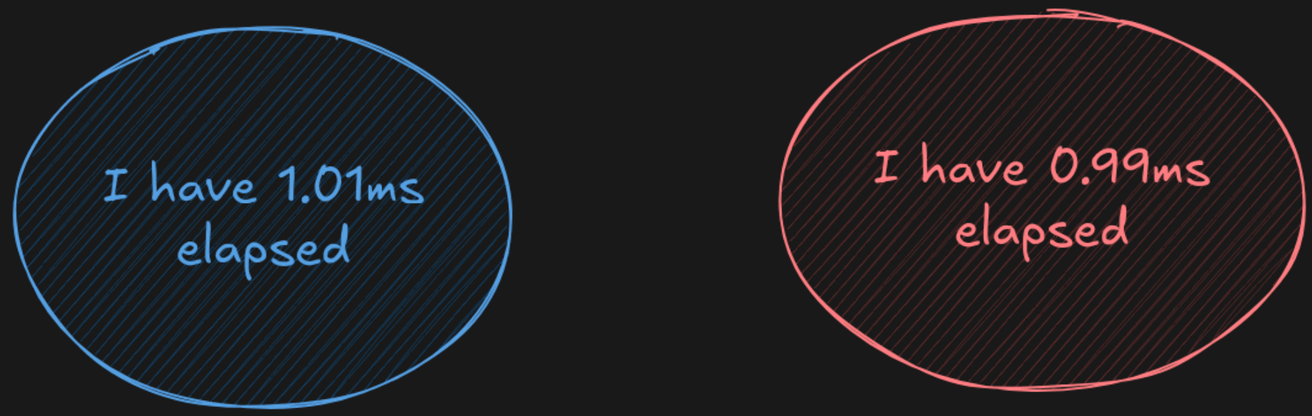
CLOCKS

- Devices derive time from a clock
- A repeating signal with a fixed frequency
- Generated by a quartz crystal



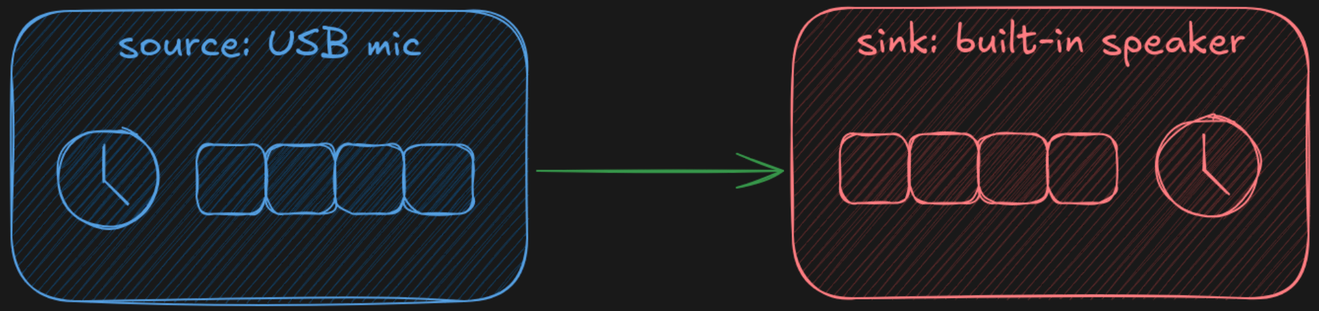
TIME IS AN ILLUSION

- Different devices will have different crystals
- With *slightly different* notions of time



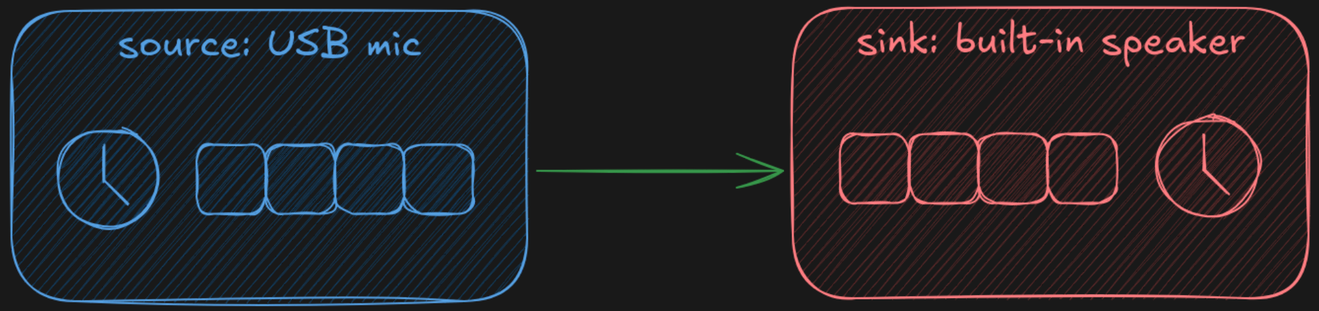
What does 1ms even mean?

THE RATE MATCHING PROBLEM



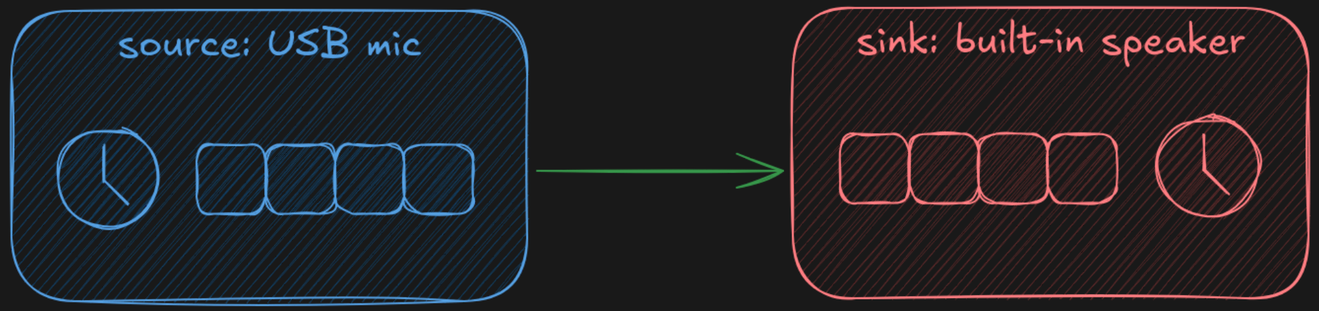
- Two devices with different clocks
- Producing data every “1ms”, say

THE RATE MATCHING PROBLEM



- What happens if the source is *slower* than sink
- Or if the source is *faster* than the sink?

THE RATE MATCHING PROBLEM



- Source slower than sink: *Underflow!*
- Sink slower than source: *Overflow!*

GSTREAMER

GSTREAMER: CLOCKS

- Clocks are represented via `GstClock`
- Report the current time
- Wait until a certain time

GSTREAMER: CLOCKS

- One for the `pipeline`
- Elements synchronise to this
- i.e. decide when to render data

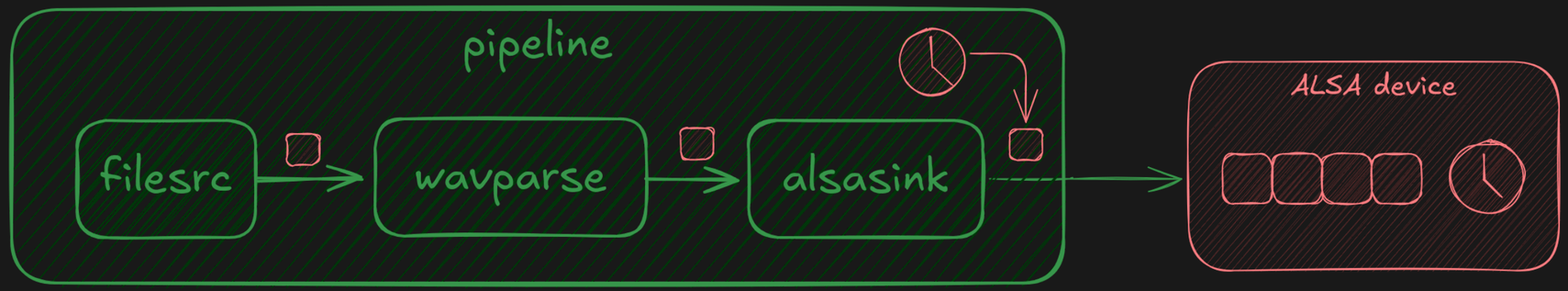
GSTREAMER: CLOCKS

- Elements can provide their own clocks
- System clock
- Audio clocks
- NTP, PTP, netclock

GSTREAMER: CLOCKS

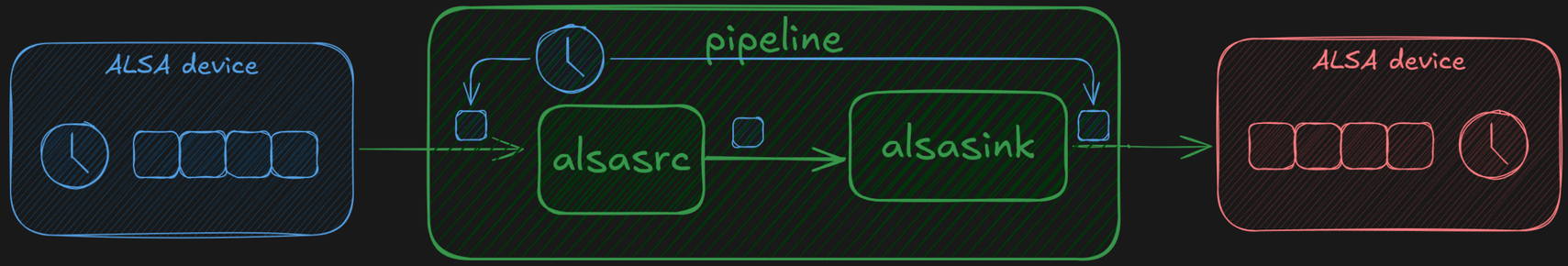
- Pair to system clock for continuous output
- Linear regression
  - $t^1_{audio} \implies t^1_{sys}$
  - $t^2_{audio} \implies t^2_{sys}$
  - $\therefore t^{now}_{sys} \implies t^{now}_{audio}$
- With some heuristic filtering

GSTREAMER: CLOCKS



A non-live pipeline

GSTREAMER: CLOCKS



A live pipeline

GSTREAMER: RATE MATCHING

- Only some elements sync to clock
- `GstAudioBaseSink`
- `GstAudioBaseSrc`
- `GstAudioAggregator`

GSTREAMER: RATE MATCHING

They may…

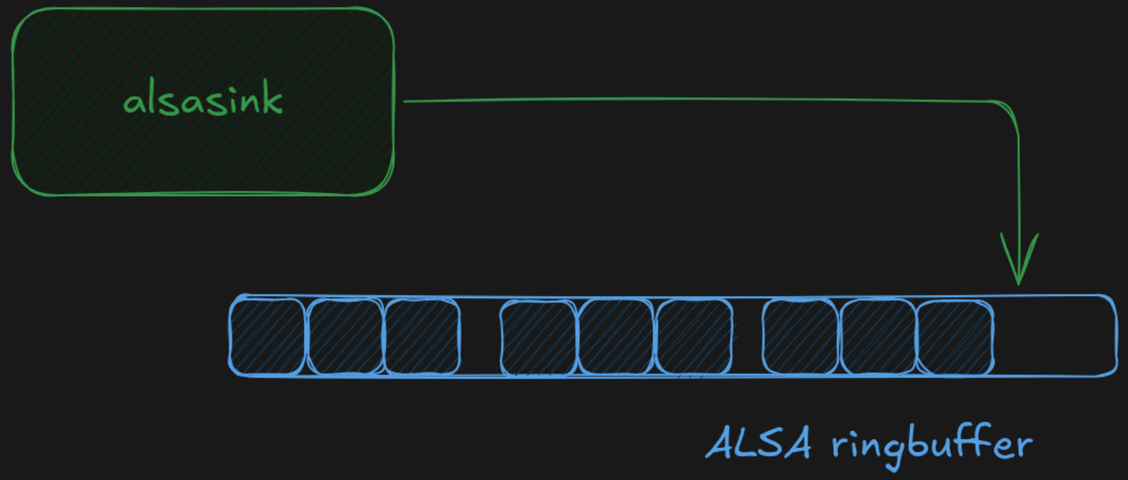
1. Skew
2. Resample
3. Custom

GSTREAMER: RATE MATCHING

- Skewing is most common
- “Spread” out the error
- Less audible glitches

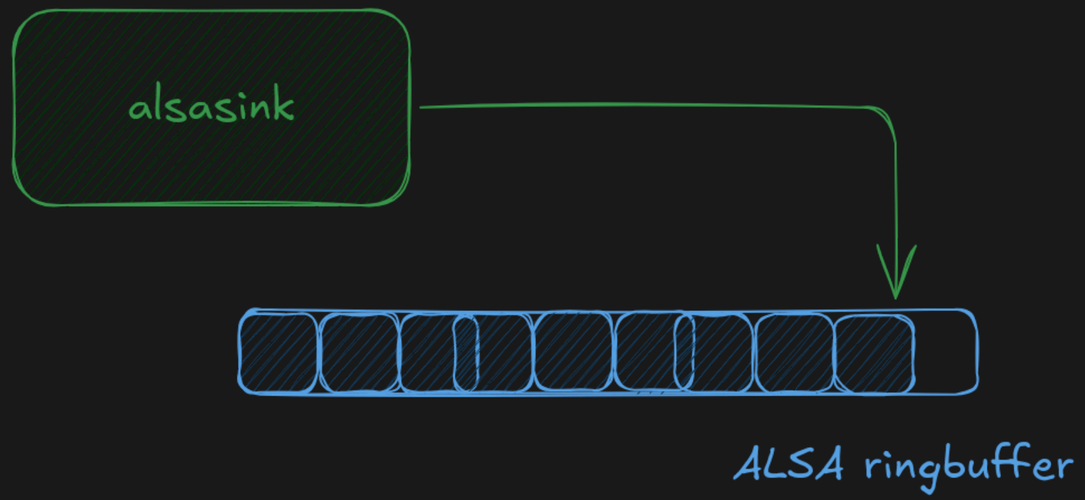
GSTREAMER: RATE MATCHING

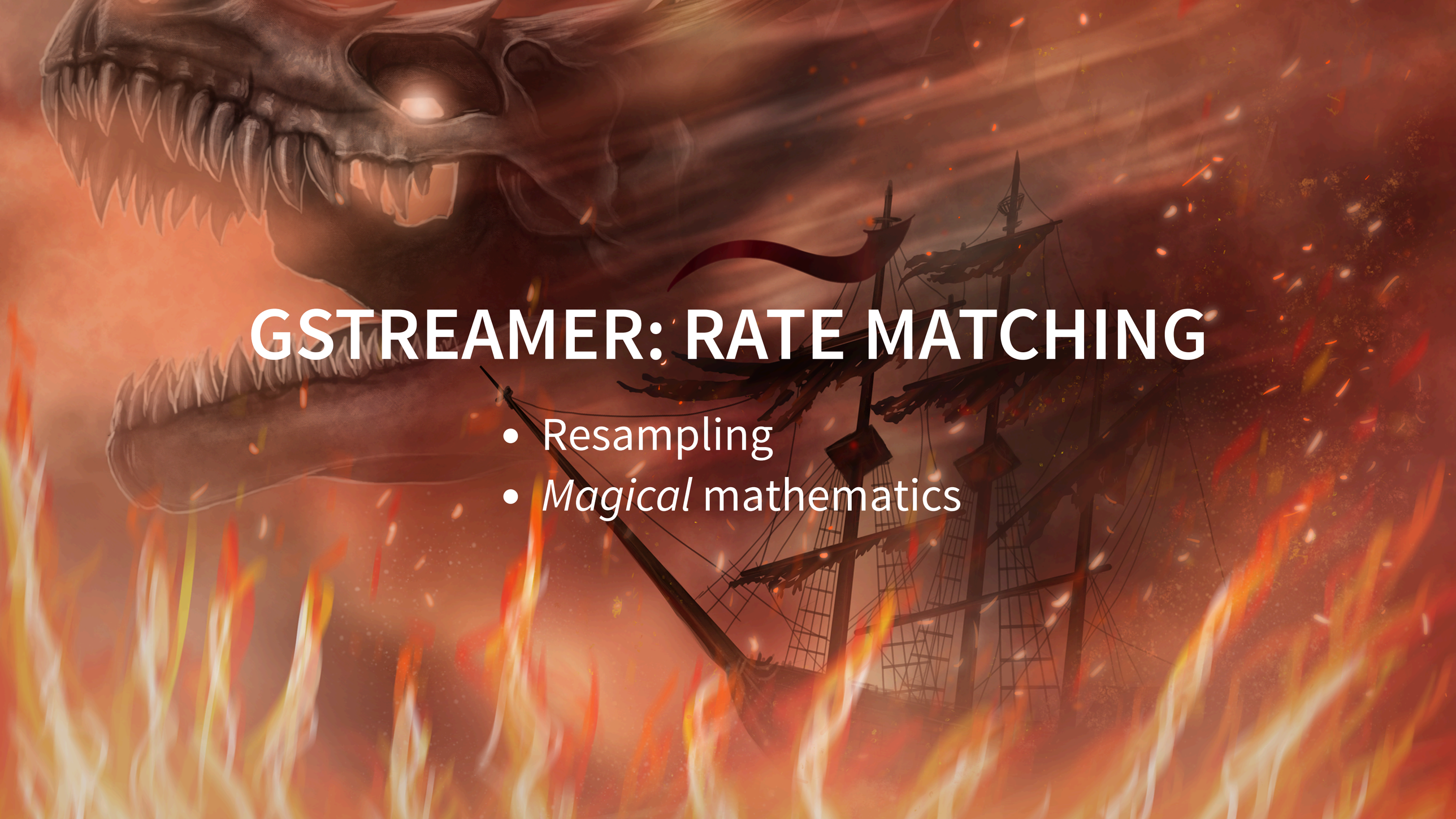
- Sink faster than source
- Insert smaller gaps instead of underflowing



GSTREAMER: RATE MATCHING

- Sink slower than source
- Overwrite smaller chunks instead of overflowing





GSTREAMER: RATE MATCHING

- Resampling
- *Magical* mathematics

GSTREAMER: RATE MATCHING

- Produce the correct number of samples
- Implemented as linear interpolation
- Naïve and low quality
- Limitations of `GstAudioRingBuffer`

GSTREAMER: RATE MATCHING

- Custom
- Let the application do *something*
- Like adjust the audio clock itself
  - “Pull” a PLL

PIPEWIRE

PIPEWIRE: MODEL

- Also a graph
- `process()` cycle
- Quantum-sized buffers
- Much like JACK

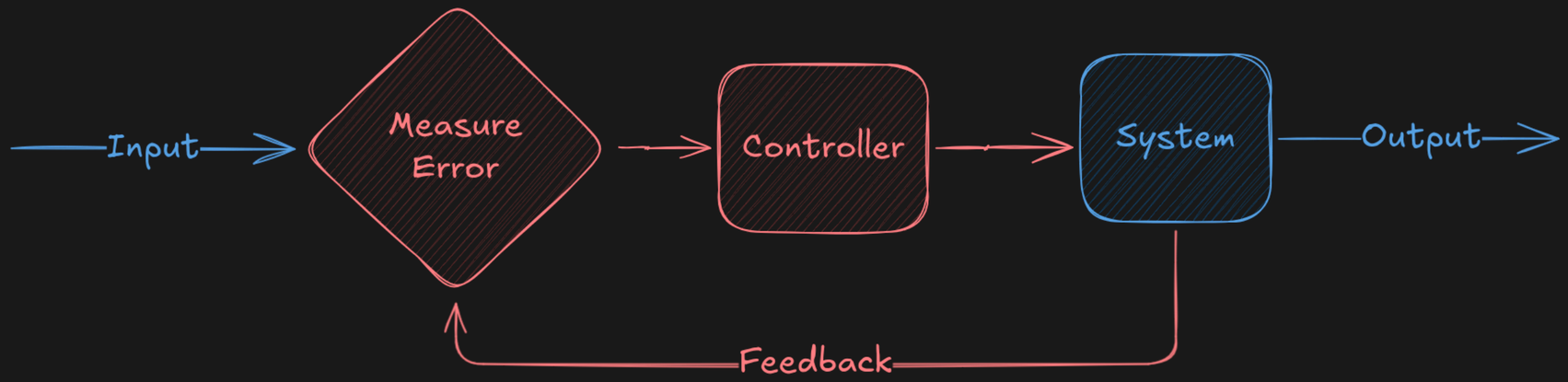
PIPEWIRE: DRIVERS

- One *driver* per graph
- All other nodes are *followers*
- Provides tick-per-quantum
- May or may not participate in data flow

PIPEWIRE: DRIVERS

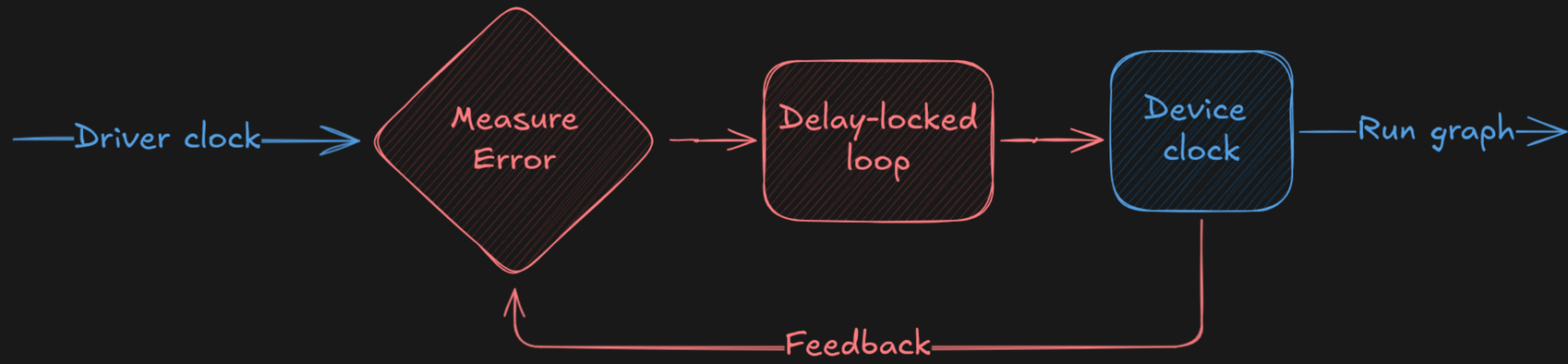
- System clock
- PTP
- Devices

DETOUR: CONTROL SYSTEMS



Closed loop control

PIPEWIRE: DELAY-LOCKED LOOPS

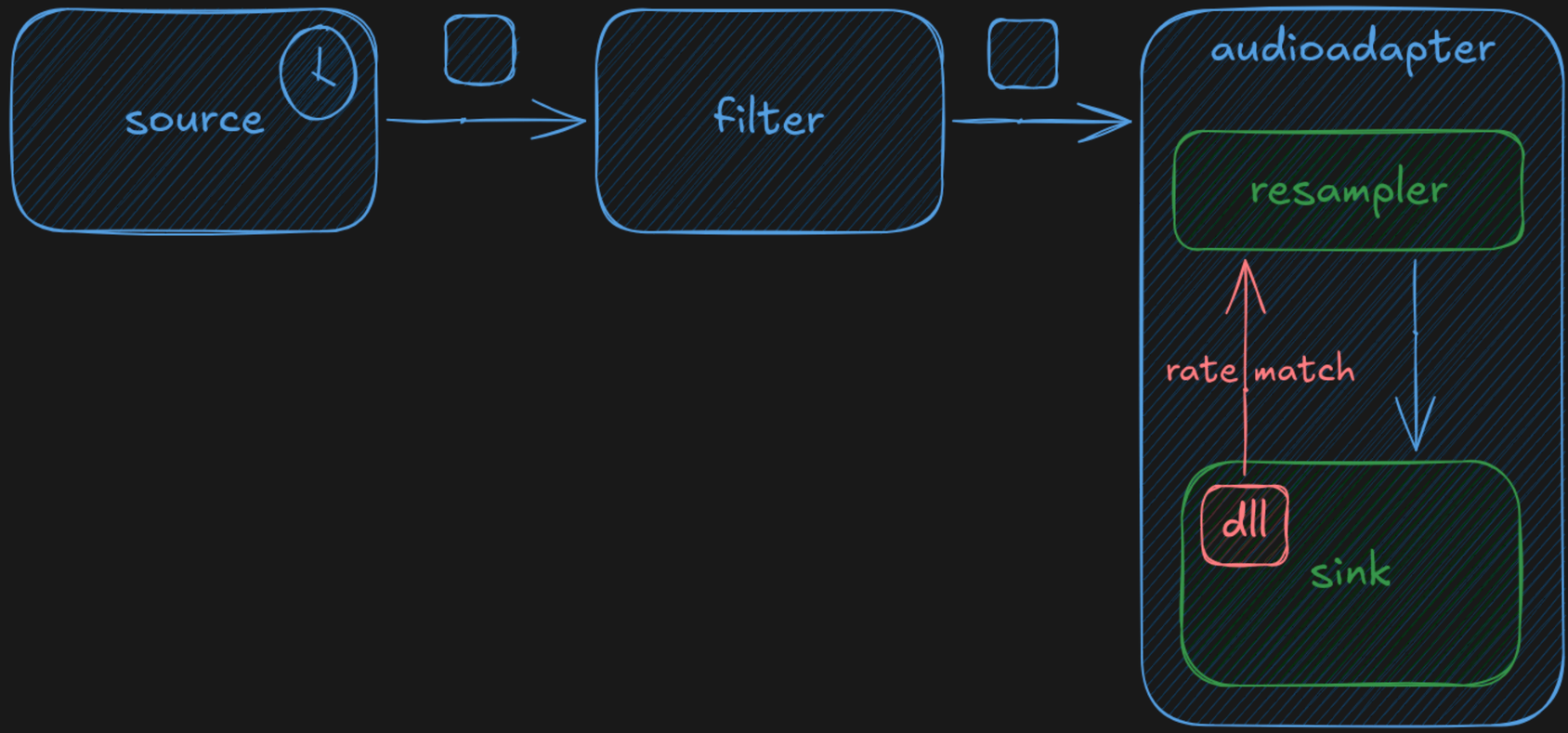


Rate matching as a control system

PIPEWIRE: DELAY-LOCKED LOOPS

- For each cycle, we have:
- `nsec`: Current cycle start time
- `next_nsec`: Expected next cycle start time
- *The relative rate of the driver and device clock*

PIPEWIRE: RATE MATCHING



PIPEWIRE: RATE MATCHING

- Can adapt to arbitrary clocks
- Can also adapt across graphs & mismatched quanta
- Resampler does fine-grained adjustment
- Converges quickly, hovers in a tight range
- Can use other mechanisms too

PIPEWIRE: RATE MATCHING REFERENCES

- Heavily inspired by JACK
- Bunch of work by Fons Adriaensen
  - Using a DLL to filter time
  - Controlling adaptive resampling

OBSERVATIONS

- System clock as a source for continuous time
- Open-ish (linear regression + heuristics) vs. closed control loops
  - Audio clocks work well
  - Network clock jitter can be problematic

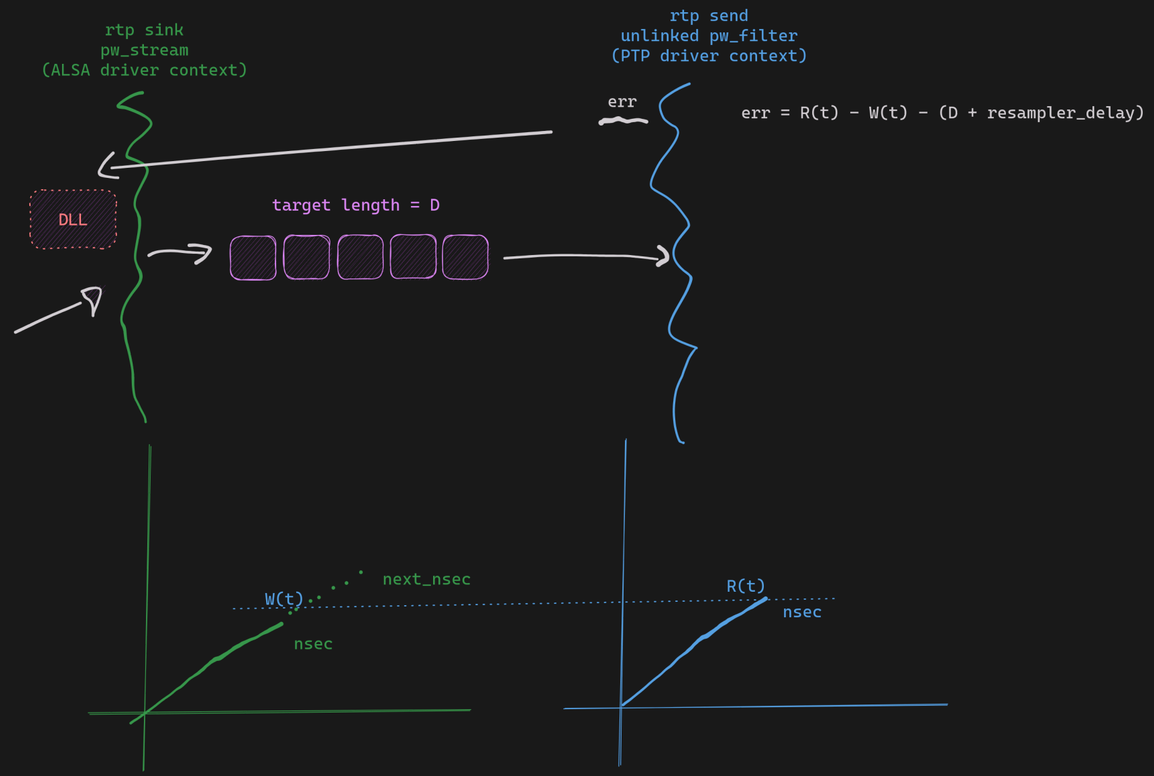
OBSERVATIONS

- You can skip `GstAudioRingBuffer`, like
  - Decklink elements
  - `pipewireaudiosink`
  - `pulsedirectsink`

QUESTIONS?

asymptotic

BONUS SLIDE



cross-graph rate matching