Time and Synchronization for dummies

Edward Hervey

edward@collabora.com

bilboed@bilboed.com

Goals

- Feel comfortable with timing and synchronization in GStreamer
- From real-life examples ...
- .. to what GStreamer can do ...
- .. and understand why

gstreamer

Time

- Ordering (past, present, future) of events
- Measure of duration of events and intervals between them
- “a certain number of repetitions of one or another standard cyclical event … constitutes one standard unit of time such as the second”

gstreamer

Chrono-meter (clocks)

- Sundials, water clocks (clepsydra)
- Hourglass (Magellan) and candles
- Mechanical clock
- Quartz and atomic (caesium) clocks

Time

- Clocks have different rates and precision
- Clocks measure the passing of time (duration, intervals)
- The absolute time is not useful
  - *Meeting at 232895437843294 !*
- You need a reference
  - 01/10/2013 00:00 UTC is 332895437840000
  - Meeting in 10mins from "now"

GstClock

- API and base implementation
- Monotonic rate
- Get current clock ***absolute time***
  - gst_clock_get_time()
- Schedule event for absolute time
  - gst_clock_id_wait()/wait_async()
- Works without a pipeline

GstClock

- Different implementations
- Doesn't matter for rest of talk
- Assume it's the monotonic POSIX clock

Clock
Absolute
Time

Buffer timestamps

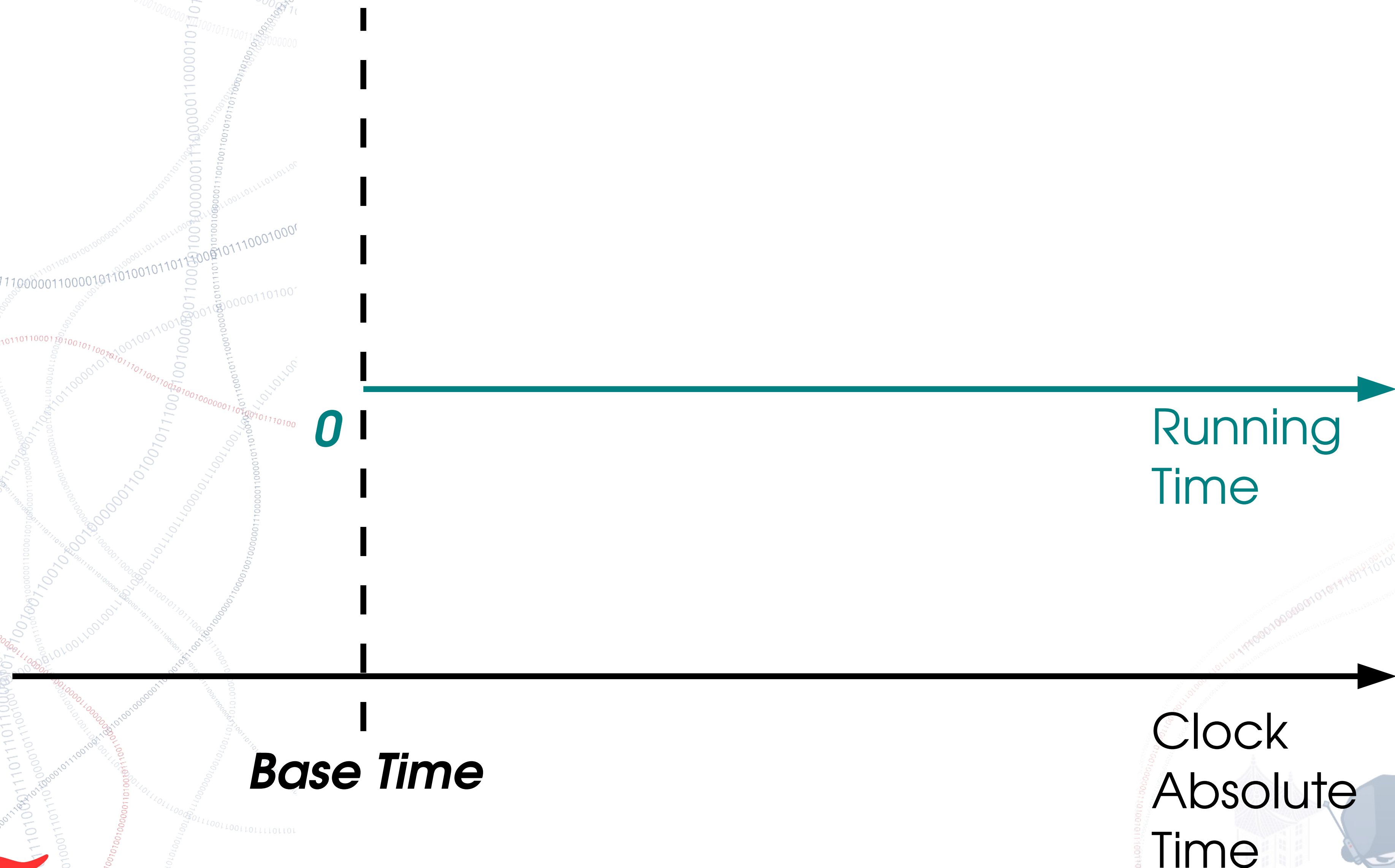
- videotestsrc ! timeoverlay ! xvimagesink sync=False
- Timestamps on buffers produced by videotestsrc
- 0, 1/30s, 2/30s, 3/30s, ...

Buffer timestamps

- Argh, everything goes too fast !
- Without synchronization, it's like unix shell piping (+/-)
- But the buffers had timestamps !
- Synchronization, it's useful (c) (tm)
- How are we going to synchronize against a clock ?

Running time

- We want buffers to be synchronized N seconds after we start playing
- Moment the pipeline switches to playing (***base time***)
  - + N seconds, AKA : ***Clock running time***

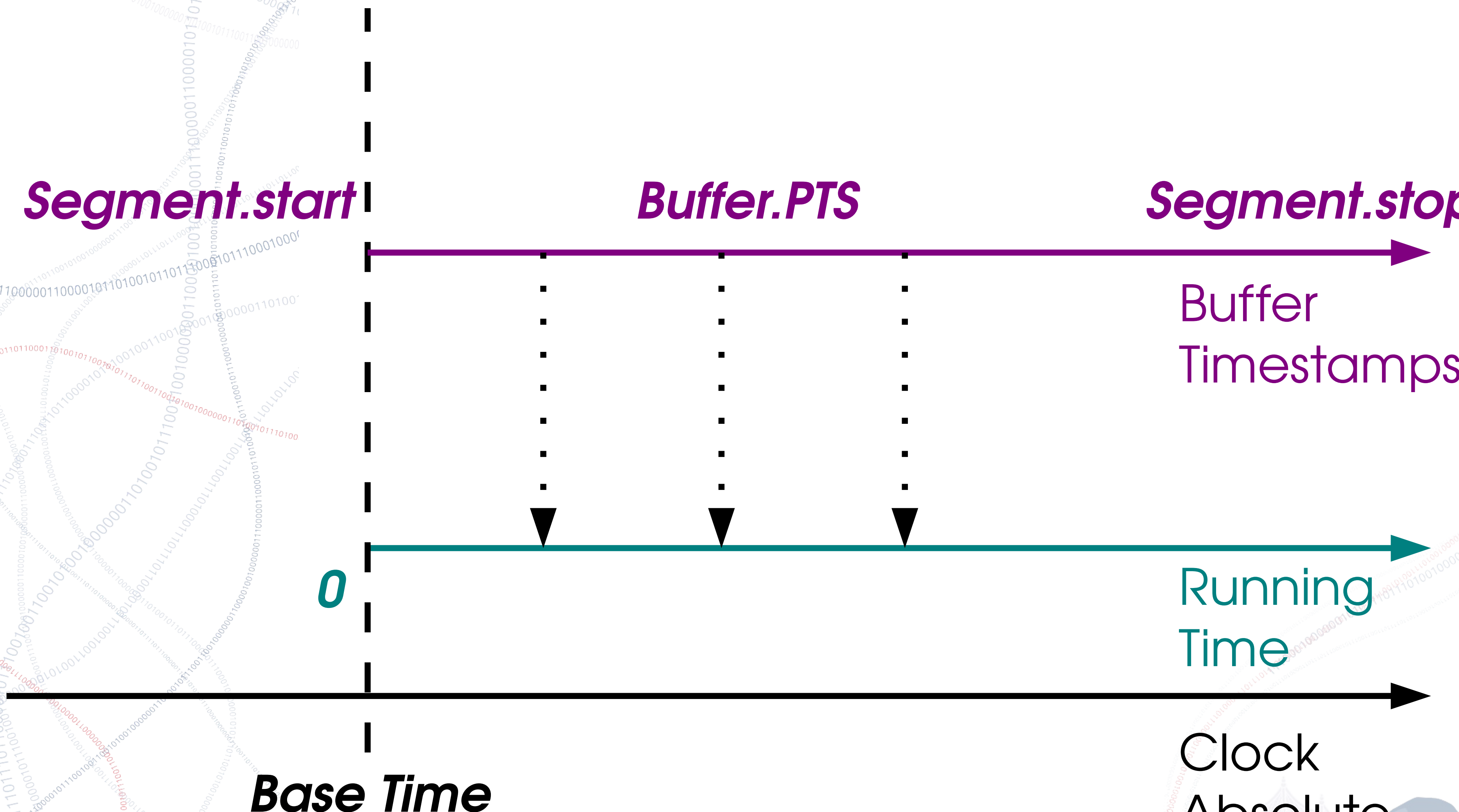

0
Running Time
Base Time
Clock Absolute Time

Segment

- How do we figure out the running time for buffers ?
  - Take the absolute value ? What if the first buffer PTS is not 0 ?
- We need a reference (from which to calculate the running time of each buffer)
- Enter GstSegment !

Segment

- Helps define the various time relationships in a stream
- ***start,stop***: first and last valid buffer timestamp
- For any buffer:
  - PTS – Segment.start => buffer running time
  - (not final formula)



Segment.start
Buffer.PTS
Segment.stop
Buffer
Timestamps
0
Running
Time
Base Time
Clock
Absolute
Time
gstreamer

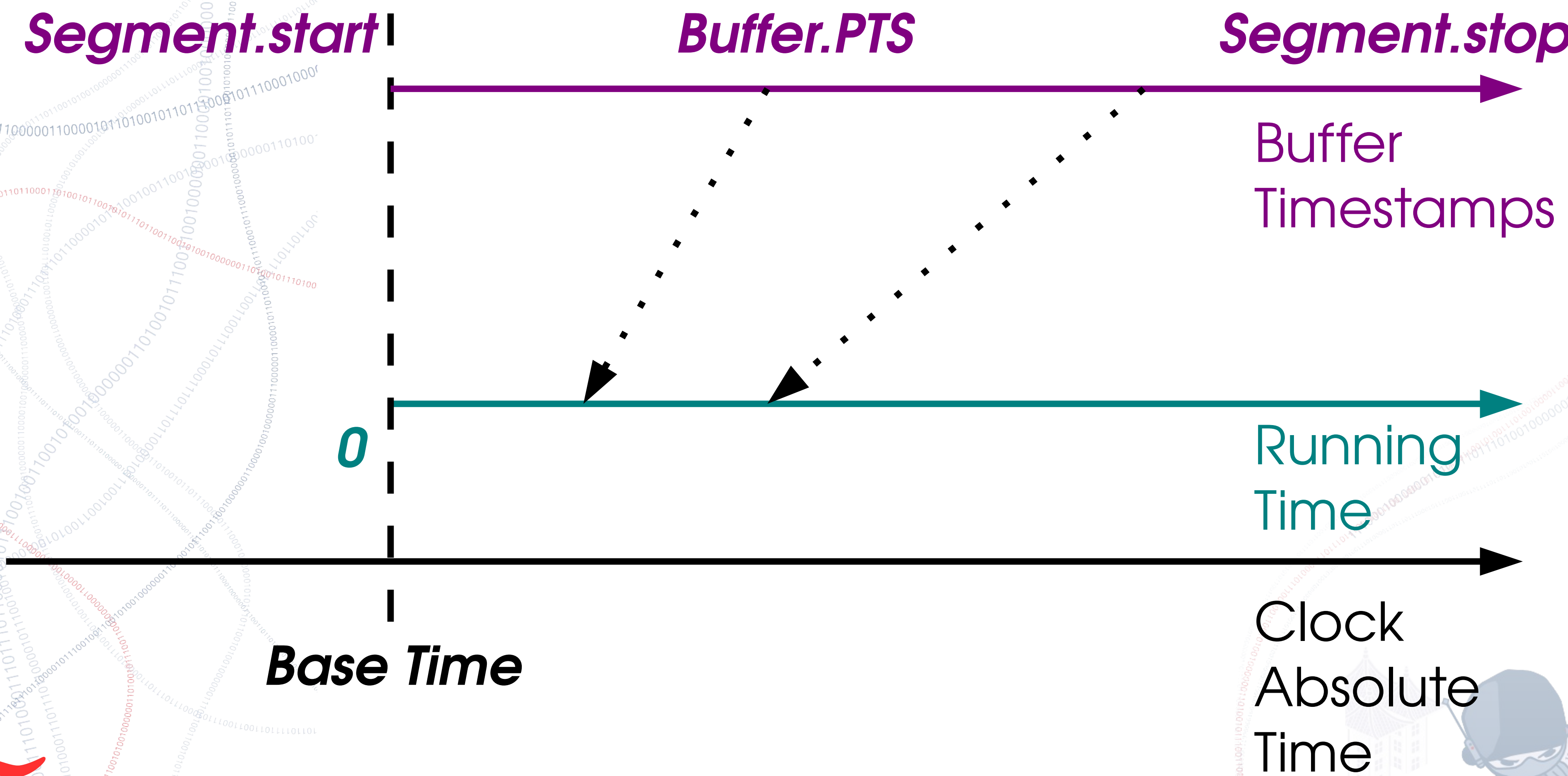
Base Time

- What if I pause ?
- Running time is the amount of time “spent” in PLAYING
- Need to update ***base_time***
- PLAYING=>PAUSED : remember running_time
- PAUSED=>PLAYING : base_time = current_absolute_time – running_time.

Segment rate

- What if I play faster/slower ?
- I want buffers to be synchronized faster/slower
- Segment ***rate*** property
- running_time gets adjusted accordingly
- (B.PTS – S.start) / ABS(S.rate)

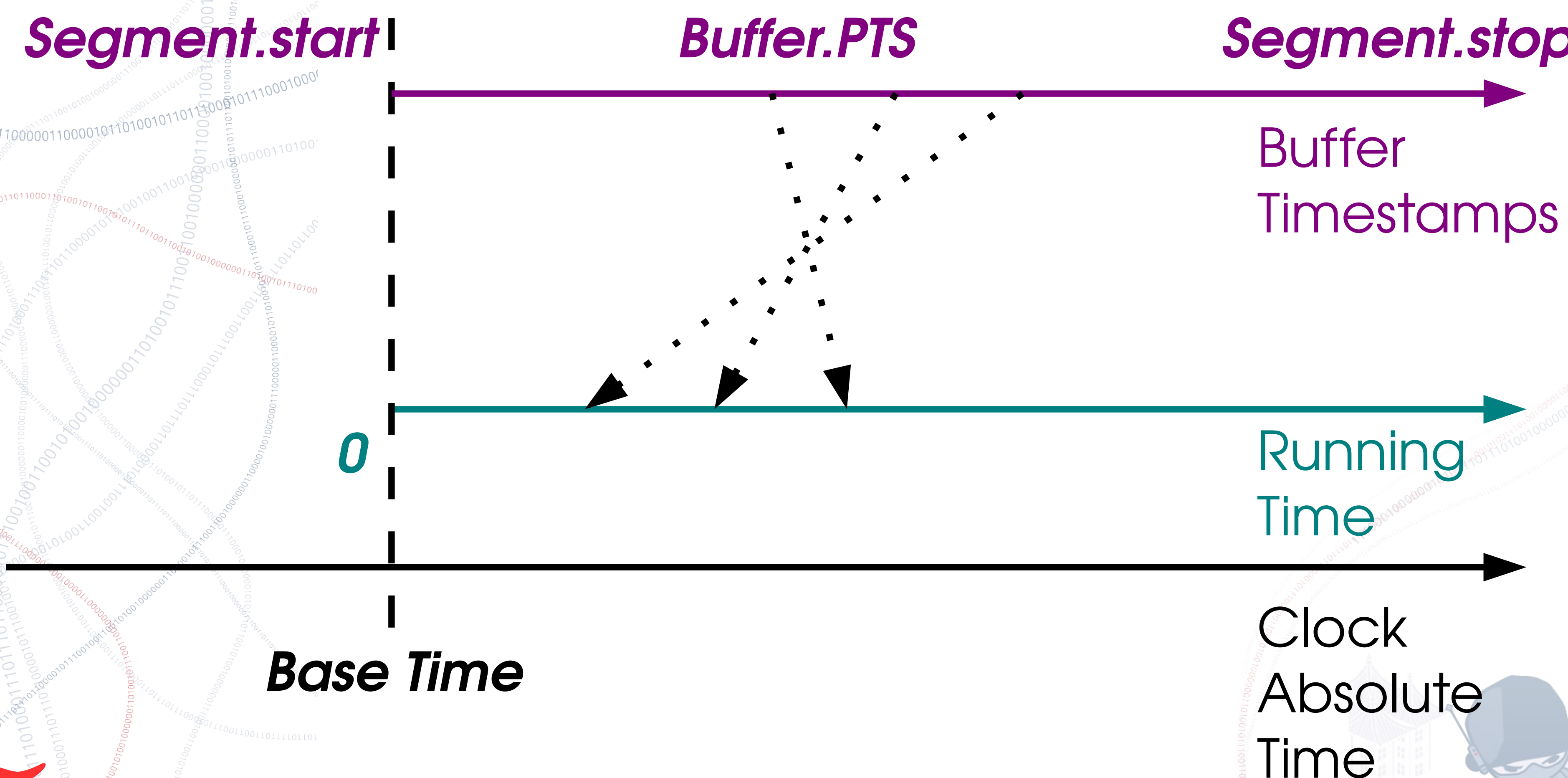
Segment.rate > 1.0



Segment rate

- What if I play backwards (in reverse) ?
- Segment.rate < 0.0
- Buffer have decreasing timestamps
- Running time is calculated using Segment.stop
  - (S.stop - B.PTS) / ABS(S.rate)

Segment.rate < 0.0



Stream time

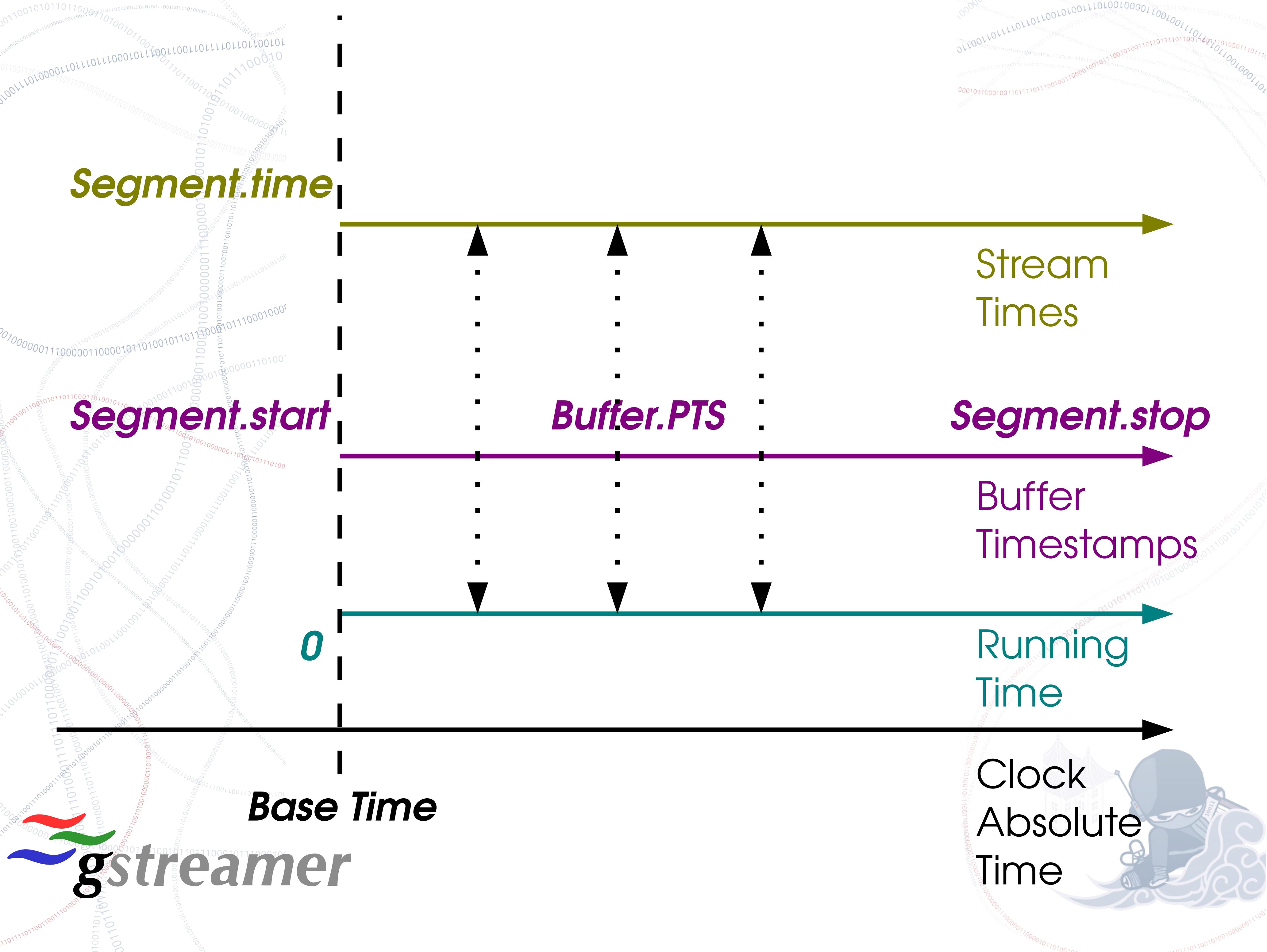
- “User-facing time”
  - Position reporting
  - Seek values
- Quite confusing since it's quite often the same as buffer time.
- When isn't it the same ?
  - RTP use-cases
  - DVB use-cases
  - Some formats

gstreamer

Stream Time

- You connect to a live presentation via RTP which started 30mins before
- RTP timestamps (i.e. Buffer timestamps) can be anything
- You want to be shown how much in the presentation you are
- => Stream Time
- ***Segment.time*** (reference for stream time)

gstreamer


Segment.time
Stream Times
Segment.start
Buffer.PTS
Segment.stop
Buffer Timestamps
0
Running Time
Base Time
Clock Absolute Time
gstreamer

So far....

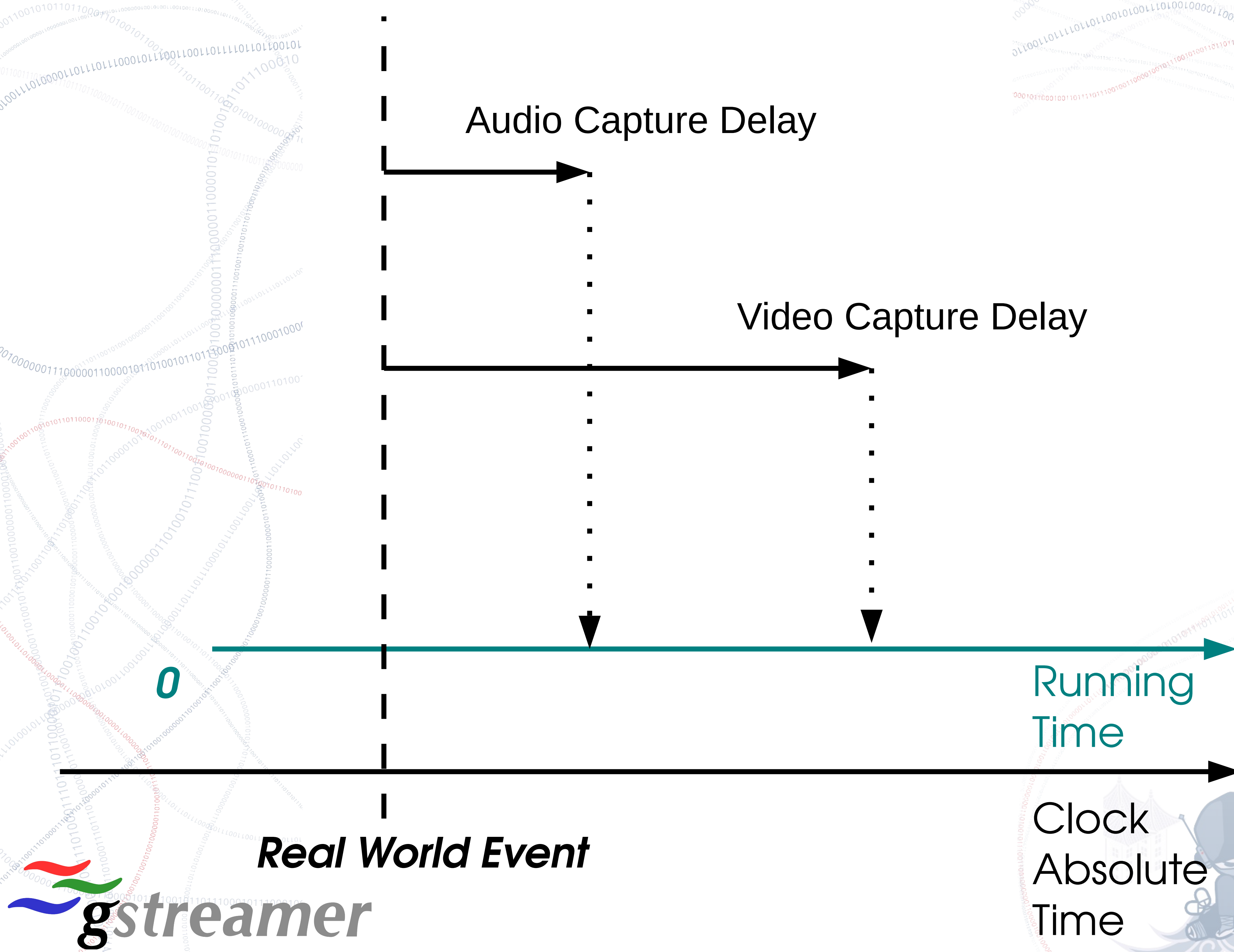
- Absolute Clock Time
- Running Time
- Buffer Time
- Stream Time
- Need base_time and segment

Live sources/pipelines

- Time and Clocks are not just used for synchronizing buffers/events,
- Also used for knowing when an event happened.
- Live sources (webcam, microphone, ...)

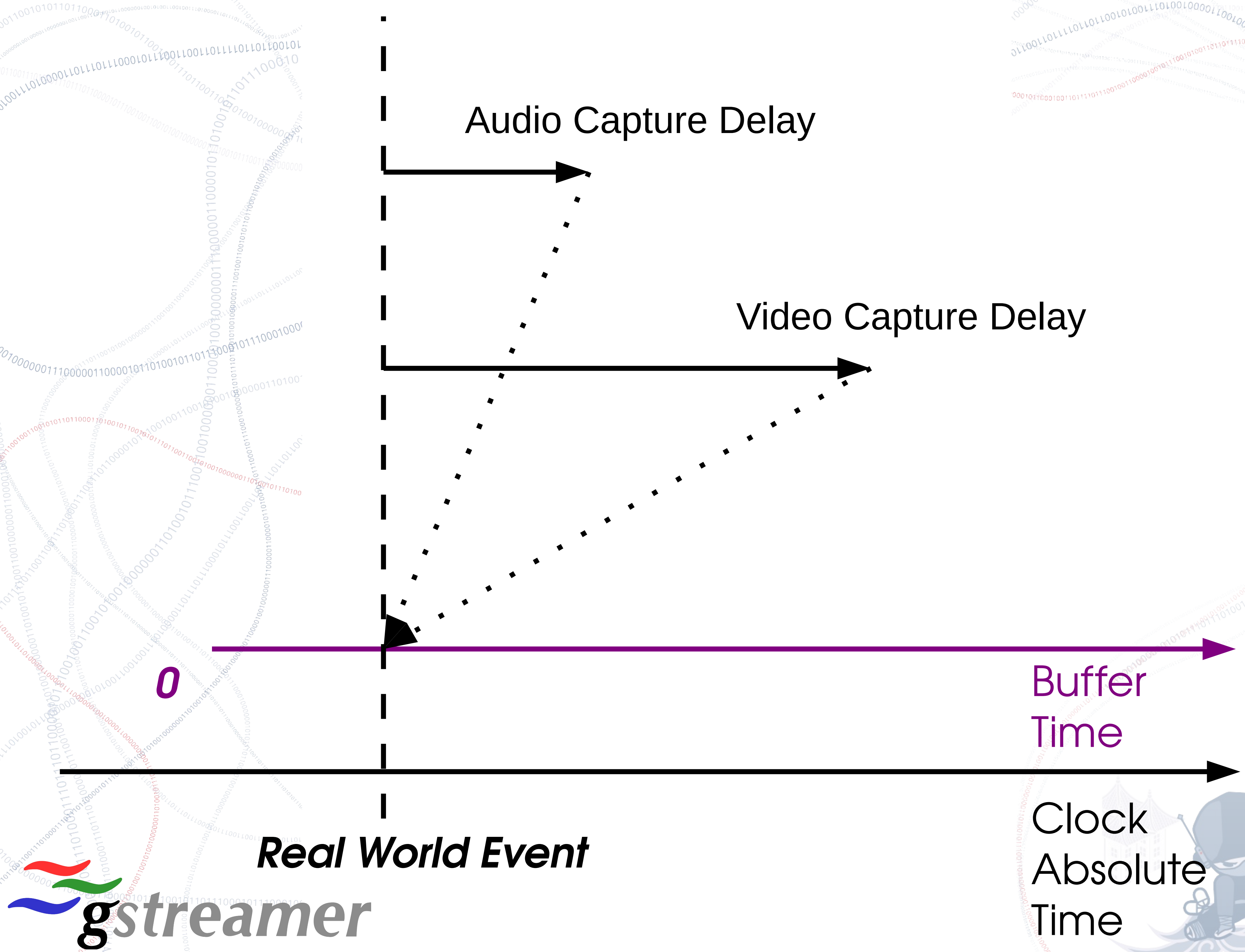
Live sources/pipelines

- A "live" event is an event that happens "now"
- If you try to capture too early/late you will miss it
- "now" is the current running time of the clock.


Audio Capture Delay
Video Capture Delay
0
Running Time
Clock Absolute Time
Real World Event
gstreamer

Live sources/pipelines

- The same event captured over different sources should have the same timestamp
- But we have different capture time (1 audio segment duration vs 1 webcam frame duration)
- So we just subtract that value from the current running time ?



Audio Capture Delay
Video Capture Delay
0
Buffer
Time
Clock
Absolute
Time
Real World Event
gstreamer

Live sources/pipelines

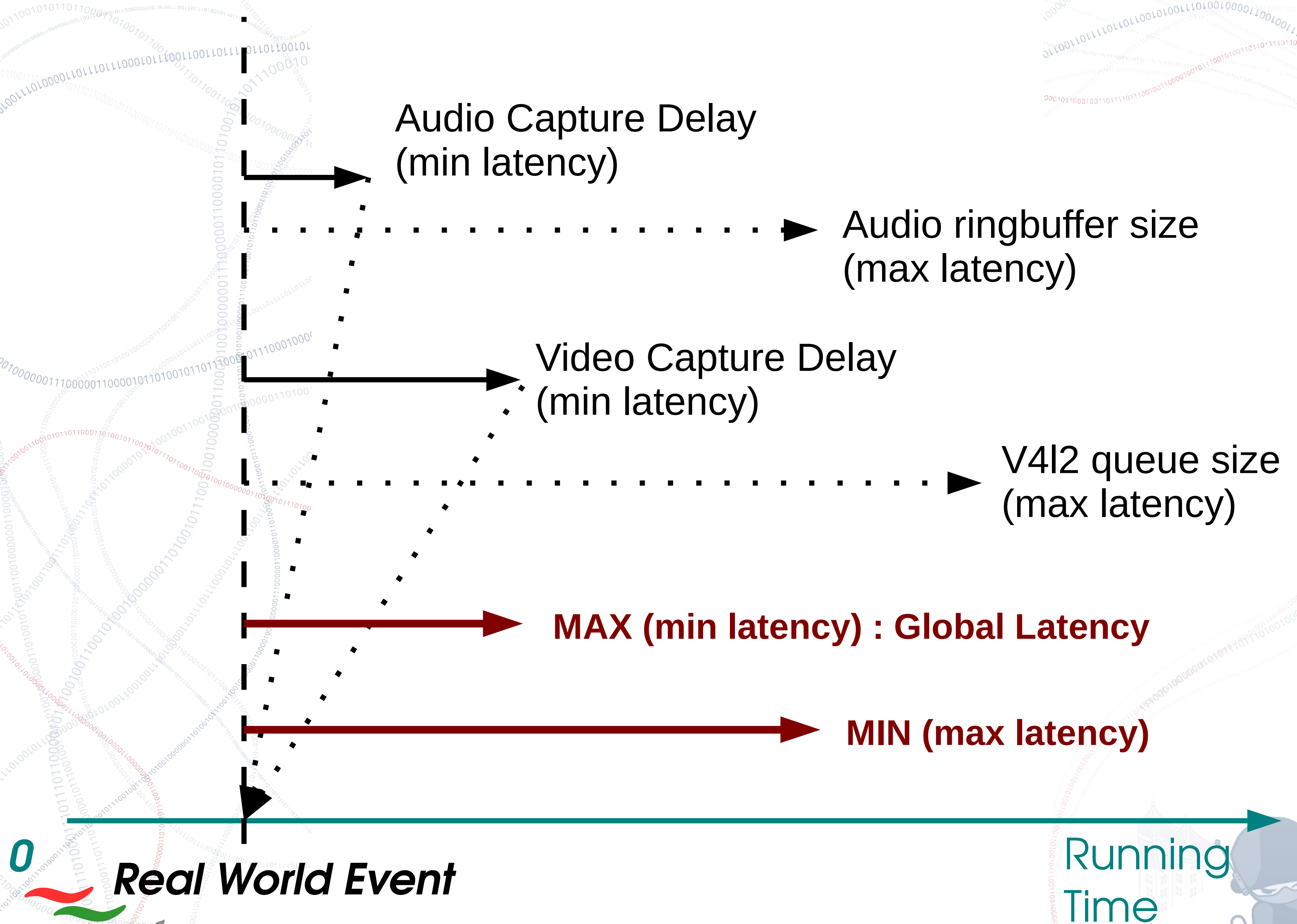
- Subtracting capture delay from running-time helps ...
- But would result in all buffers always arriving late (if you wanted to play them back in the same pipeline)
- Enter latency !

Latency

- Ensure buffers/events will be able to be synchronized downstream (i.e. Not dropped)
- As quickly as possible
- Not too early and not too late (grmbl !!!)
- How do we figure that

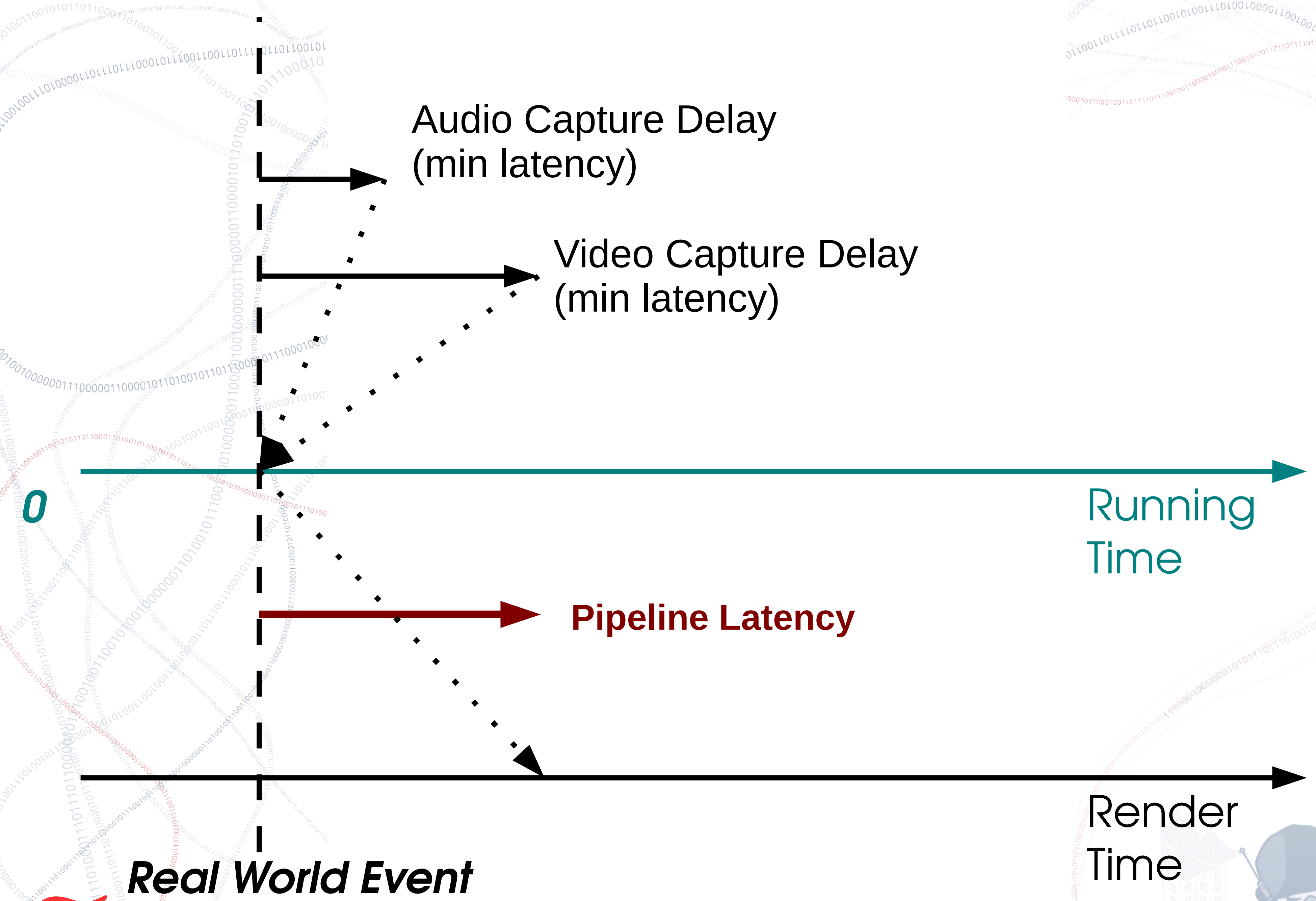
Latency

- Let every element in the pipeline report what
  - is the minimum latency it is introducing (for producing/processing data)
  - Is the maximum latency it can support (before dropping/blocking)
- GST_QUERY_LATENCY
- Pipeline emits and distributes ideal latency


Audio Capture Delay
(min latency)
Audio ringbuffer size
(max latency)
Video Capture Delay
(min latency)
V4l2 queue size
(max latency)
MAX (min latency) : Global Latency
MIN (max latency)
0
Real World Event
Running
Time

Latency

- Rendering time becomes:
- Latency + running_time


Audio Capture Delay
(min latency)
Video Capture Delay
(min latency)
0
Running
Time
Pipeline Latency
Render
Time
Real World Event

Latency

- Other elements can introduce latency
  - Decoders (frame reordering)
  - Transformation elements
  - ....
- Or increase max-latency
  - Queue !

gstreamer

No more time !

- Different clocks
- Slaving clocks and distributed synchronization
- Advanced techniques
- Go see Jan's talk
- You have the basics !

Time for questions ?

- Or time for lunch ?