Is Video4Linux ready for all cutting-edge hardware?

Ezequiel Garcia
ezequiel@collabora.com

COLLABORA

Is Video4Linux ready for all cutting-edge hardware?

COLLABORA

Is Video4Linux ready for all cutting-edge hardware?

tl;dr: no

Agenda

- Traditional V4L APIs
- New APIs
- The future

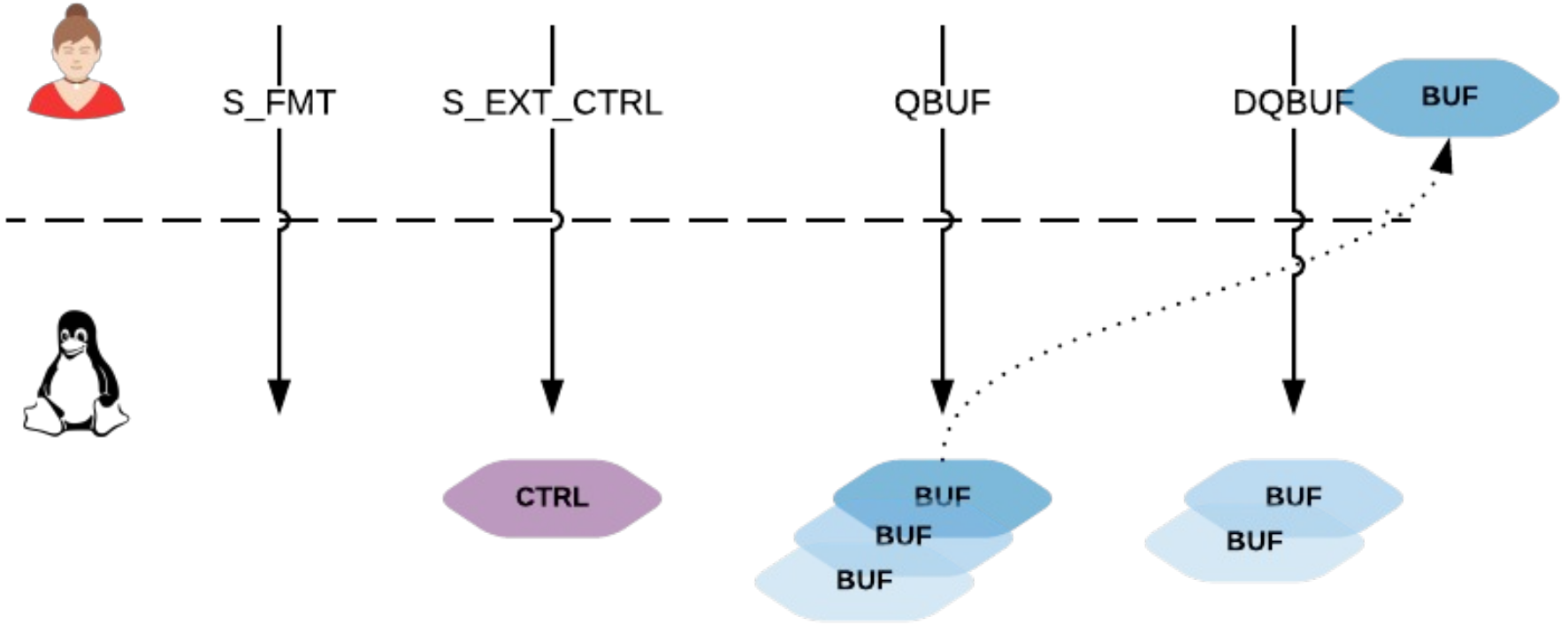
Current Video4Linux

V4L2 API

- VIDIOC_{ENUM_TRY,G,S}_FMT
- VIDIOC_{G,S}_STD, VIDIOC_QUERYSTD
- VIDIOC_REQBUFS
- VIDIOC_QBUF, VIDIOC_DQBUF
- VIDIOC_STREAMON, VIDIOC_STREAMOFF

  and more...

Stream API



Codecs

Stateful codecs

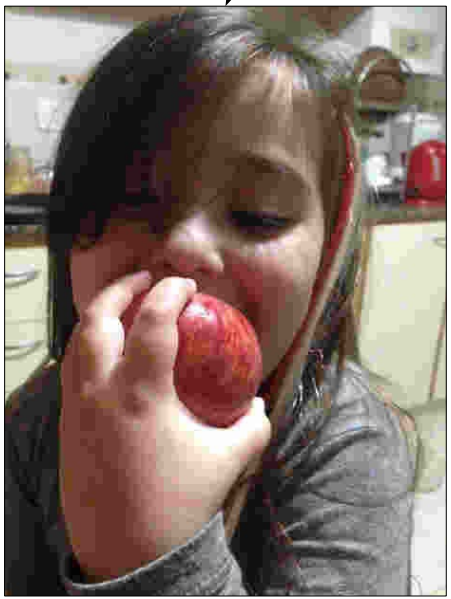
- Device handle full bitstream, so drivers shouldn’t do any parsing. Performing software stream processing, header generation etc. in the driver is strongly discouraged.
- Uses the traditional V4L (stream-based) API.
- Specification in progress:

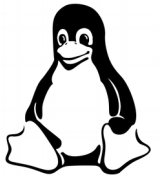
  `[PATCH 0/2] Document memory-to-memory video codec interfaces`

Stateful codecs

- Platforms with stateful codecs in mainline
  - i.MX
  - QCOM
  - Exynos
  - Mediatek
  - and more...

Stateful codecs





Headers

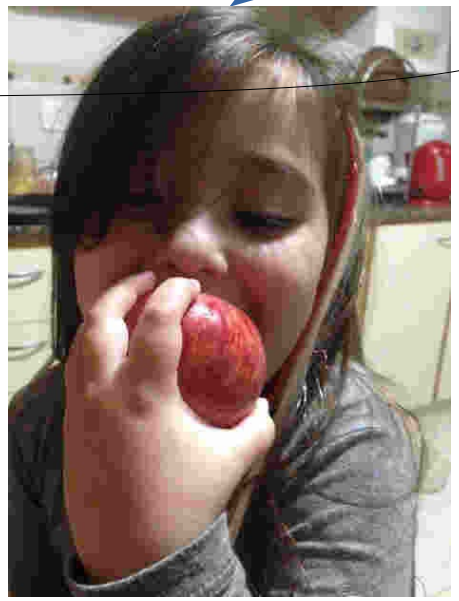
Compressed frame

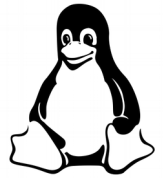
Stateless codecs

- Device accelerates the encoding/decoding job.
- Device handles raw compressed bitstreams, but needs software to do the extra parsing.
- Uses the Request (slice-based) API.
- Specification also in progress:

  `[RFC PATCH] media: docs-rst: Document m2m stateless video decoder interface`

Stateless codecs

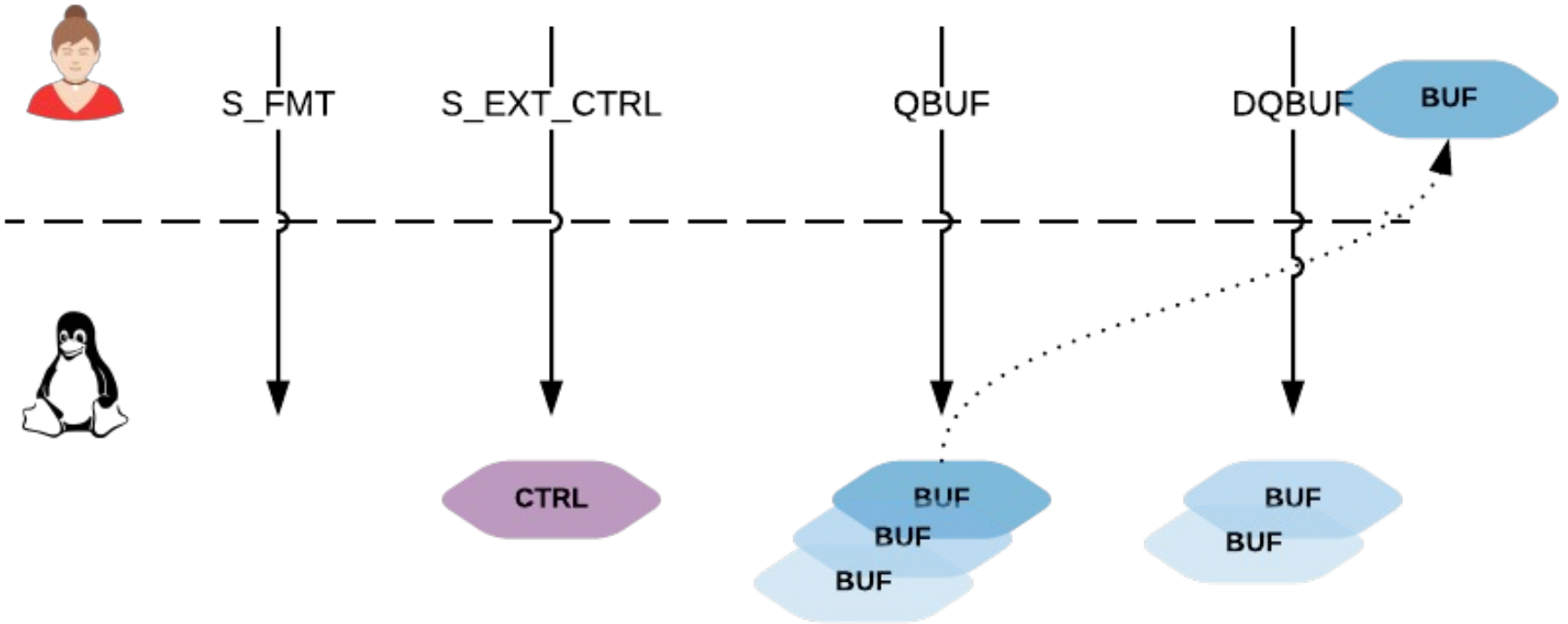




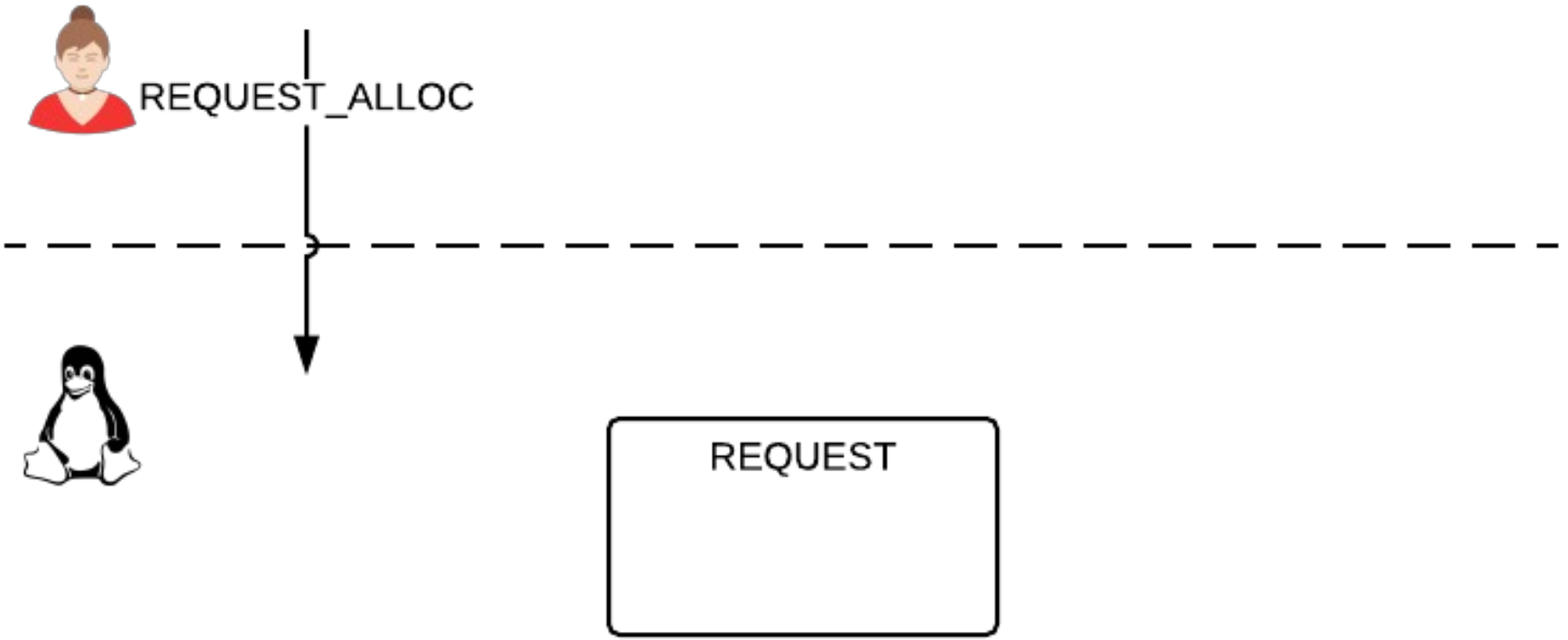
Metadata

Compressed frame

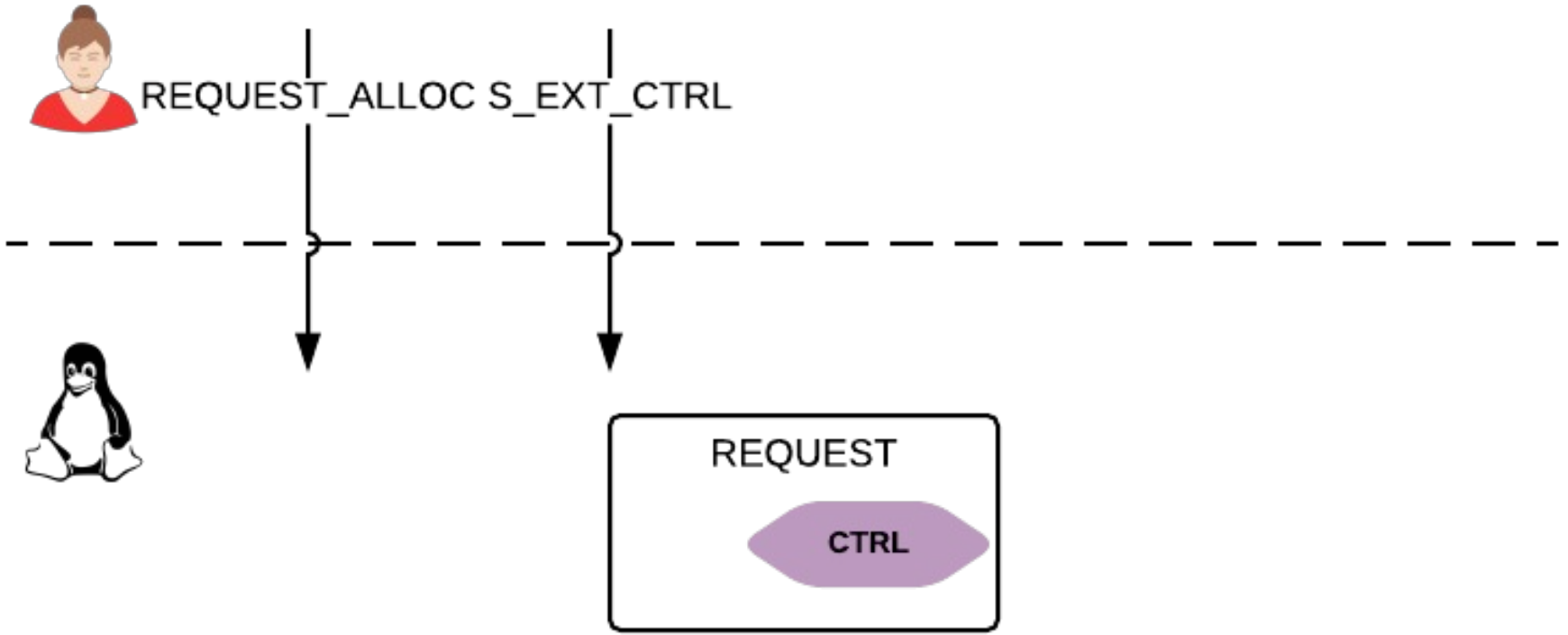
Stream API



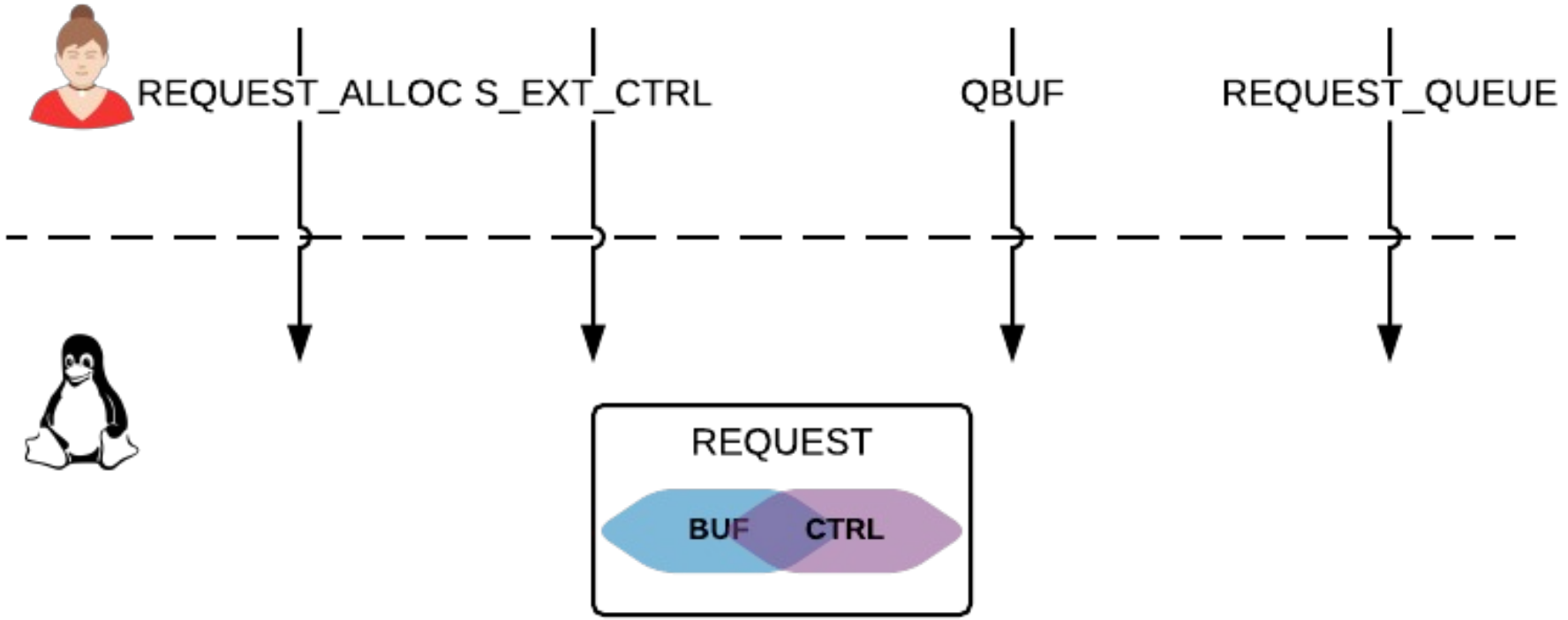
Request API

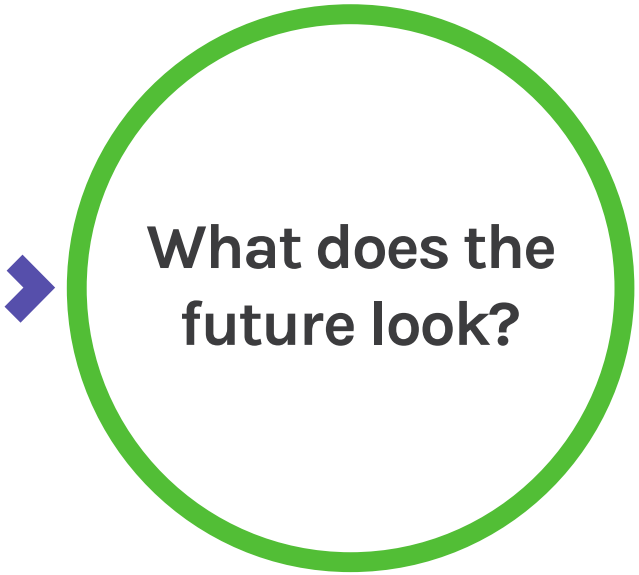


Request API



Request API


REQUEST_ALLOC
S_EXT_CTRL
QBUF
REQUEST_QUEUE
REQUEST
BUF
CTRL

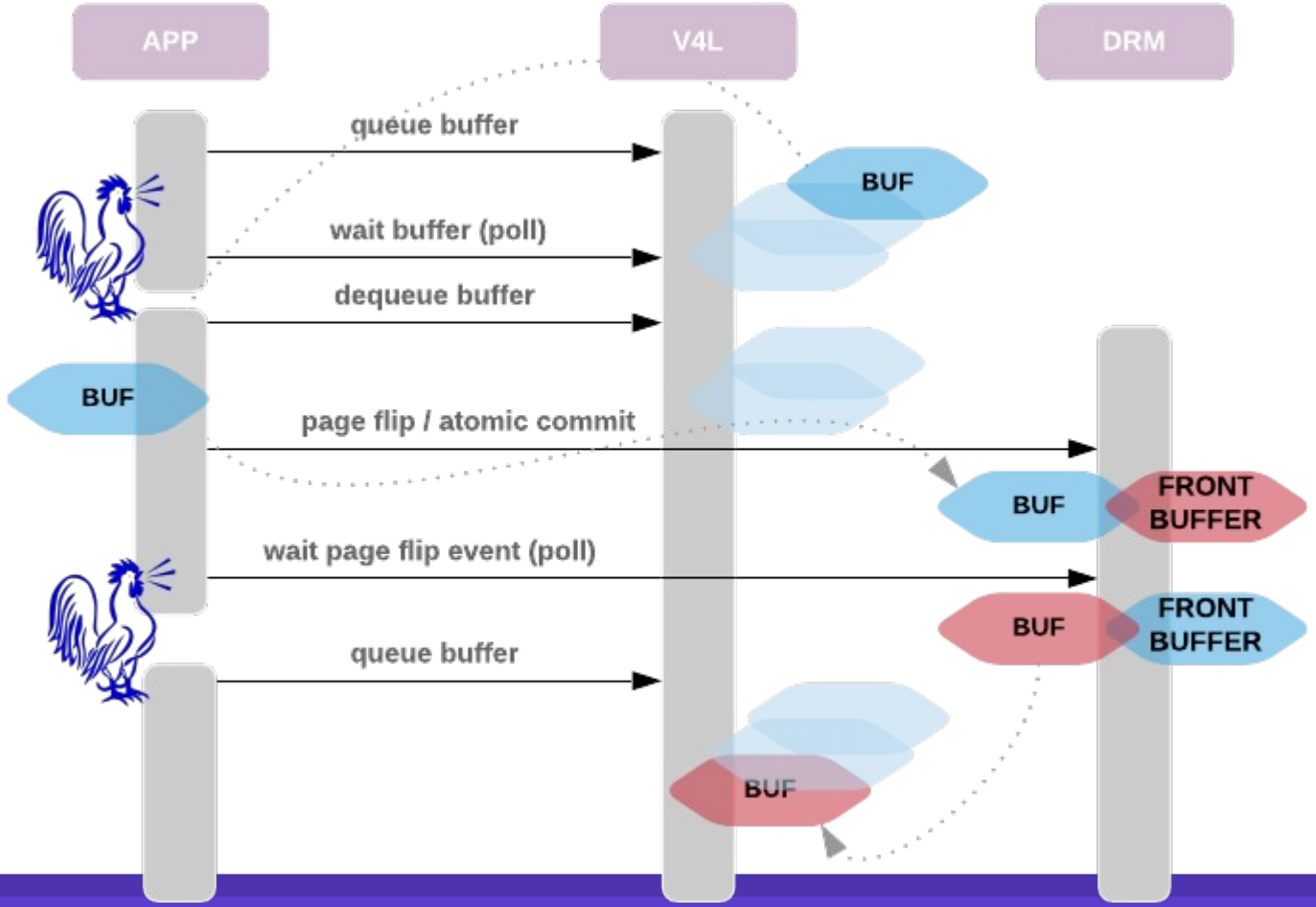

What does the
future look?

Fences

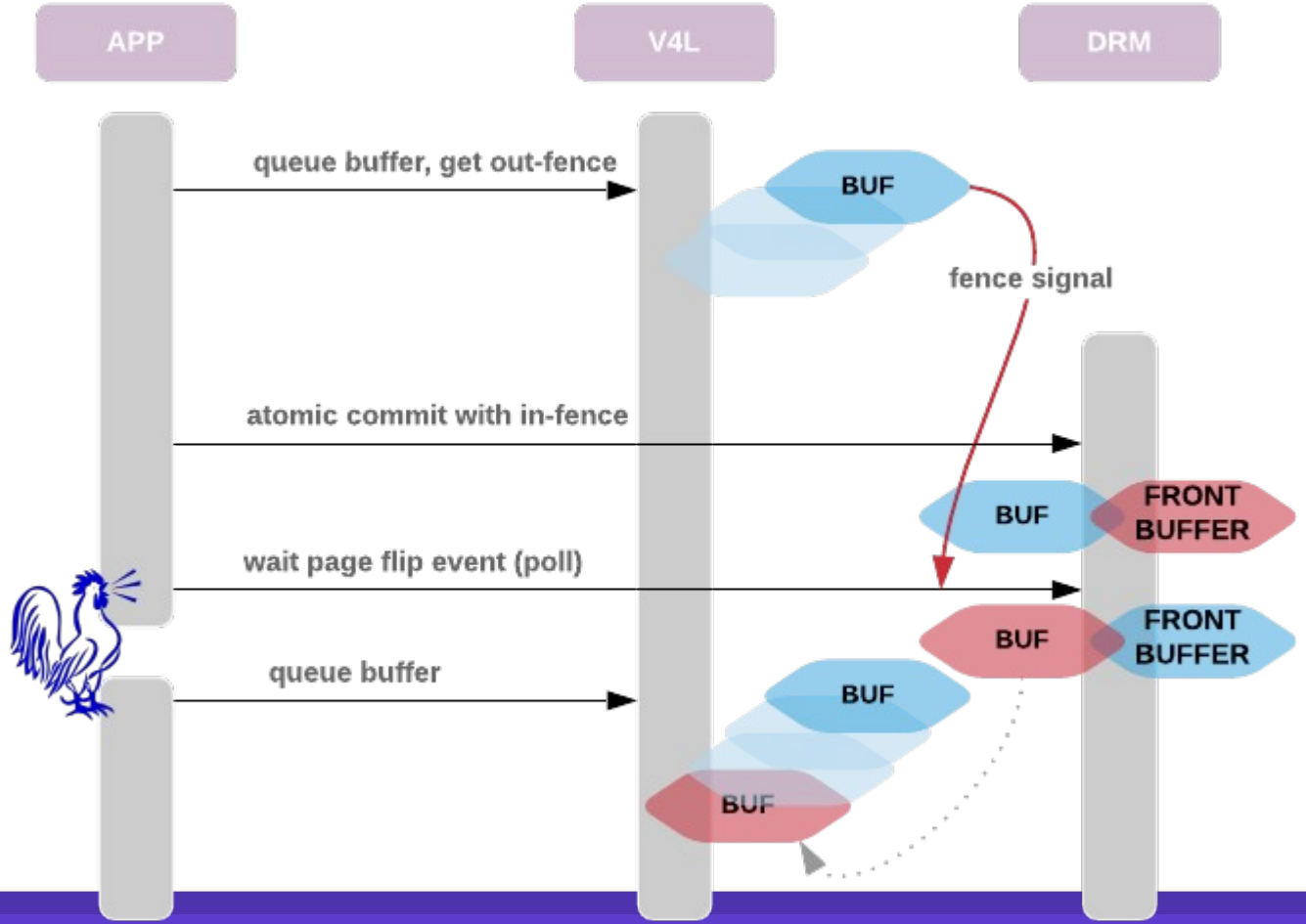
- Attaching in-fences and out-fences to buffers can reduce latency and improve efficiency.
- Work in-progress by Gustavo Padovan and Ezequiel Garcia:

```
[PATCH v10 00/16] V4L2 Explicit Synchronization
```

Without Fences



With Fences


APP
V4L
DRM
queue buffer, get out-fence
BUF
fence signal
atomic commit with in-fence
BUF
FRONT BUFFER
wait page flip event (poll)
BUF
FRONT BUFFER
queue buffer
BUF
BUF

Async UVC

- High-quality devices require more bandwidth from USB controllers and drivers.
- Multi-core SoCs capable of processing USB packets in parallel.
- Work by Kieran Bingham from Ideas on Board:

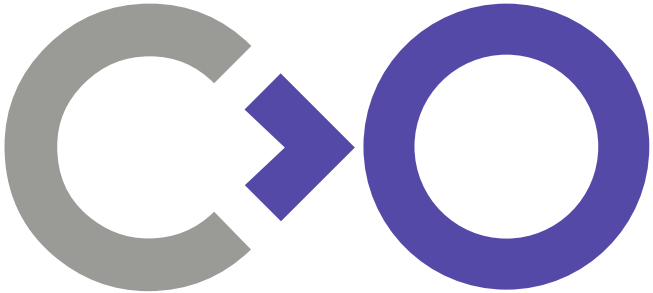
  `[RFC/RFT PATCH 0/6] Asynchronous UVC`

Async UVC (before)

```
static void uvc_video_complete(struct urb *urb)
{
        [..]

        /* copy payload */
        stream->decode(urb, stream, buf, buf_meta);

        if ((ret = usb_submit_urb(urb, GFP_ATOMIC)) < 0) {
                /* error handling */
        }
}
```

Async UVC (after)

```
static void uvc_video_complete(struct urb *urb)
{
        [..]

        /* only process headers */
        stream->decode(uvc_urb, buf, buf_meta);

        [..]
        INIT_WORK(&uvc_urb->work, uvc_video_copy_data_work);
        queue_work(stream->async_wq, &uvc_urb->work);
}
```



Thank you!