V4L2 Media Controller Request API

Karthik Poduval
Principal Software Development Engineer, Amazon Lab126

The Need for Media Request API

V4L2 Control

- Powerful control API for V4L2 Video Devices and Subdevices
- Asynchronous in design
  - SET_CTRL can be called any time
  - GET_CTRL can be called any time
- Very hard to time a SET_CTRL and GET_CTRL to specific frame
- Unable to be used as metadata
- Unable to be used to apply to specific frame

Decoder Needs

- With advent of stateless decoders, hardware becomes stateless
- State maintenance moved to user space
- User space parses bitstream and sets parameter on a per frame basis or gets per frame metadata
- Higher-level decoding libraries like
  - Vulkan Video Extensions
  - Intel VAAPI
  - GStreamer
- Drivers need to be able to support setting of decoder parameters on per frame basis or getting per frame metadata like
  - V4L2_CID_MPEG_VIDEO_BITRATE
  - V4L2_H264_SPS_*
  - V4L2_H264_PPS_*
- Parameters (controls) applicable or relevant to a given frame only.

Complex Camera/ISP Pipeline Needs

- Cameras can always benefit from per frame metadata like
  - Exposure for a given frame
  - Gain for given frame
  - Dynamic Crop Applied to a given frame
- Per frame settings can also help accomplish computational photography use cases like
  - HDR framing with low and high exposure capture
  - Multi Focal Point Capture
  - Burst Capture with varying per frame ISP settings on each

Media Request API

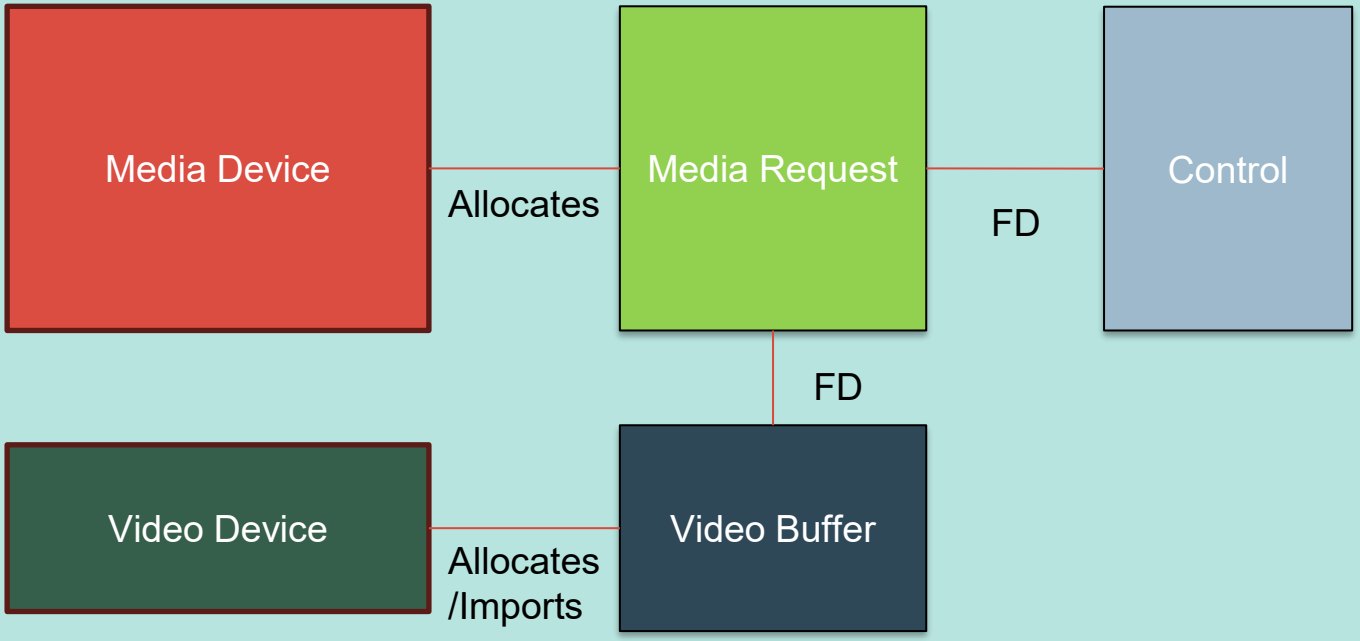
Media Controller Request API

- Designed to allow V4L2 to support devices such as
  - Stateless Codecs
  - Complex Camera ISP pipelines
- Brings in the following concepts to V4L2 Media Controller Devices
  - Per Frame Settings (VIDIOC_S_CTRL)
  - Per Frame Metadata (VIDIOC_G_CTRL)
- Without Request API media pipelines would have to
  - Flush media pipeline
  - Reconfigure new settings for next frame
  - Wait till frames arrive for new setting

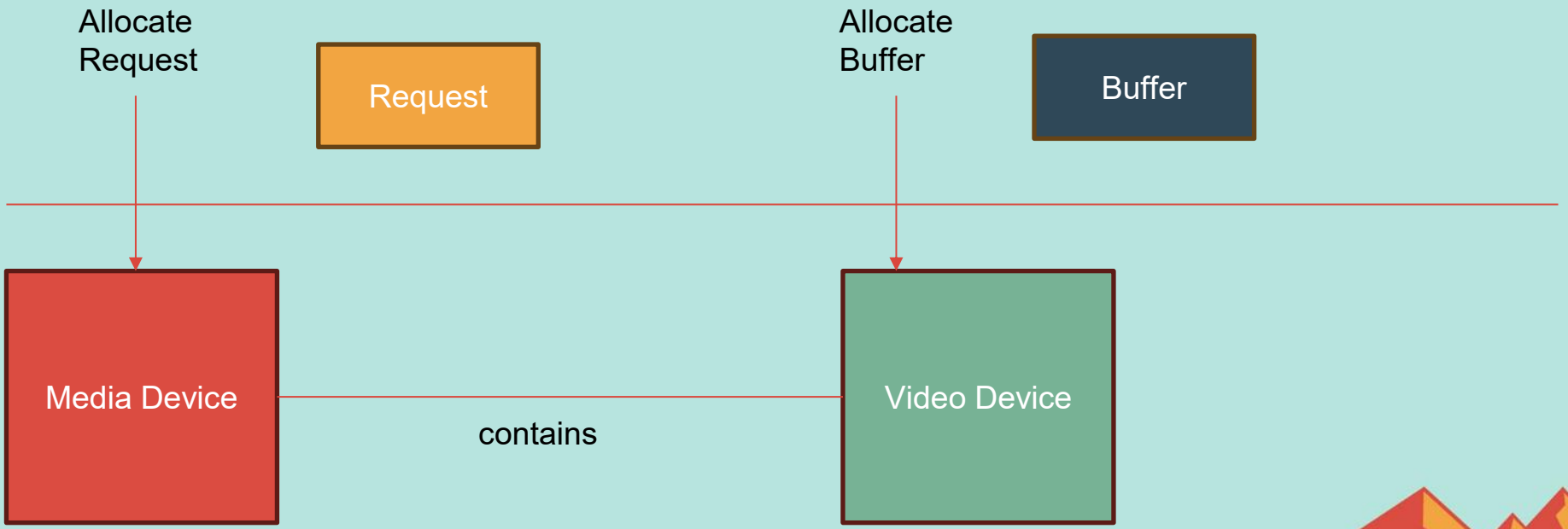
Request API Basics

- Allocate Media Request on media node
- Make control/s call on media request file descriptor
  - VIDIOC_S_EXT_CTRLS
  - VIDIOC_G_EXT_CTRLS
- Queue the Media Request object along with video buffer
- Media Driver will make sure per frame request/per frame metadata is applied in a timely manner on the video buffer (in the media pipeline)
- Once buffer returned, wait for media request to complete and query the controls to get per frame metadata

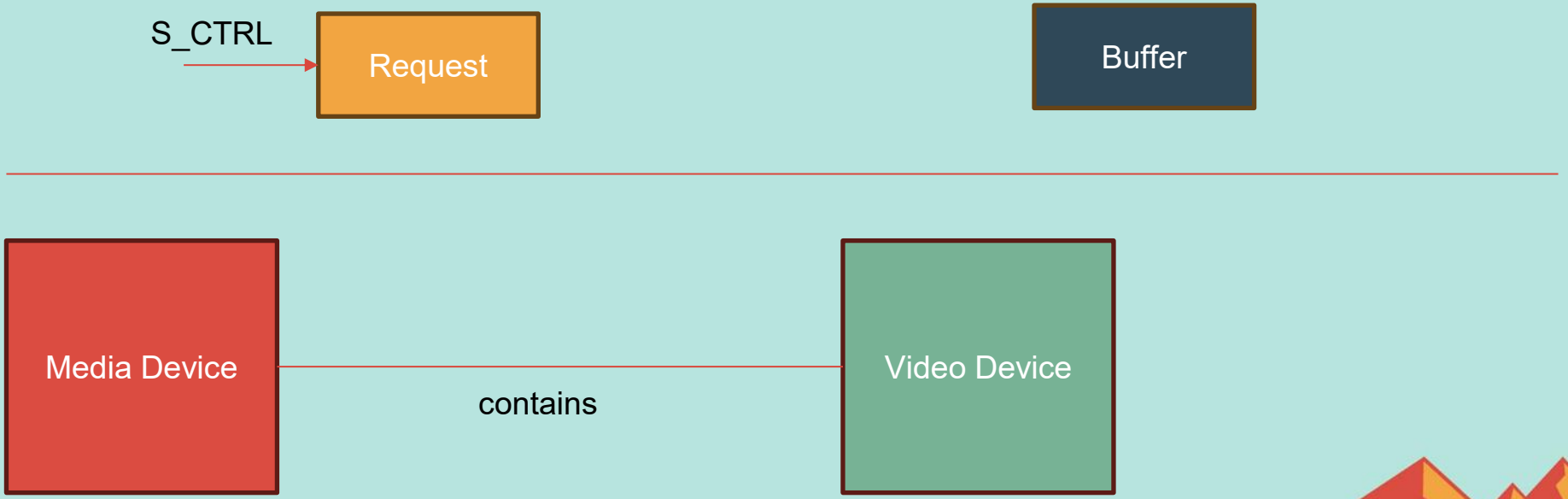
Media Request to Media Device Relationship



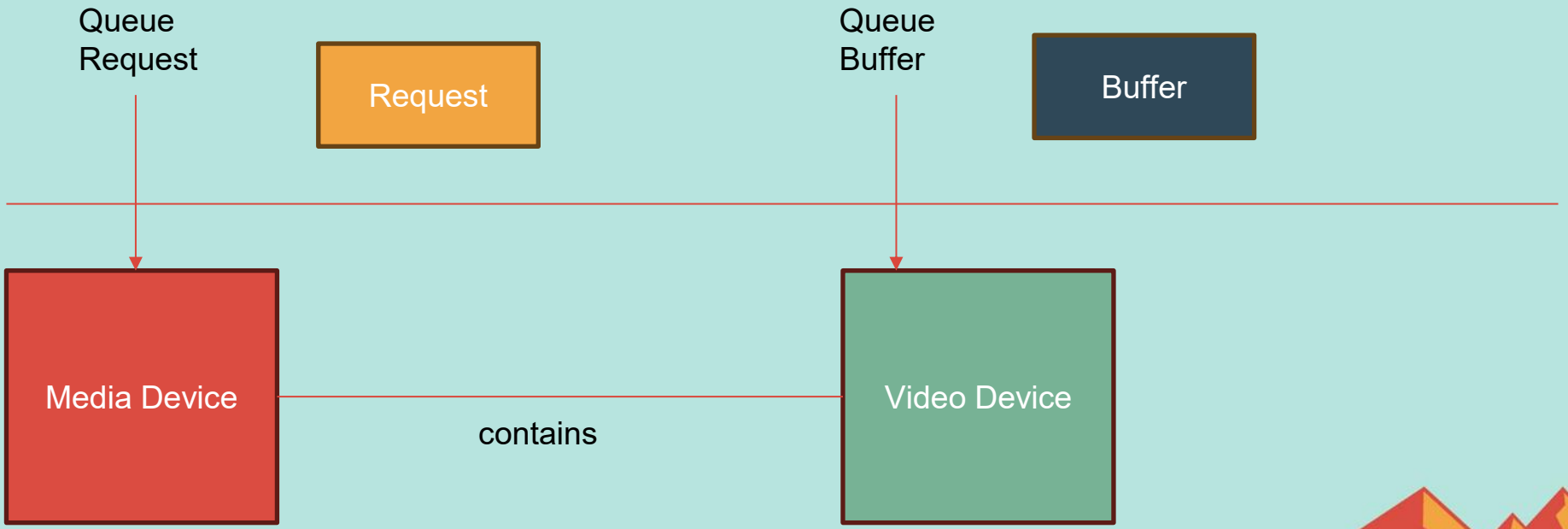
Request API Basics



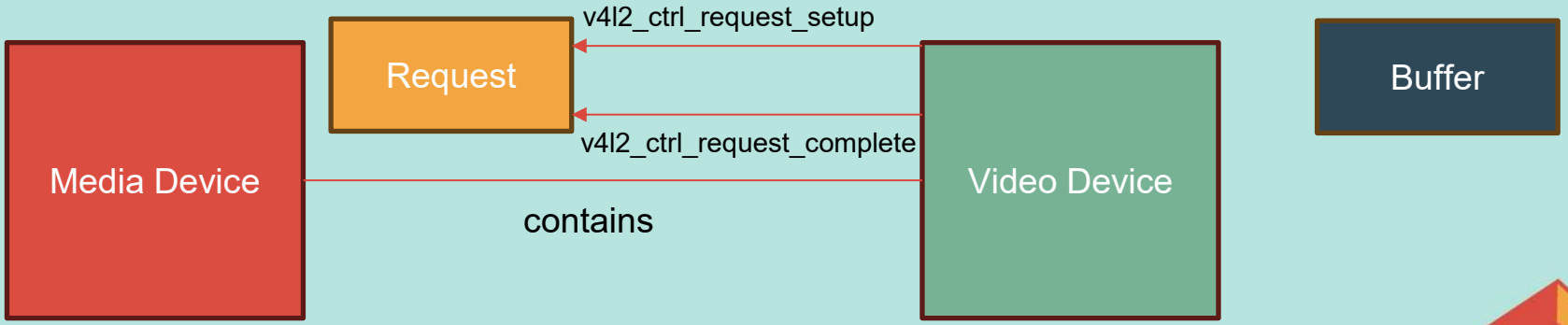
Request API Basics



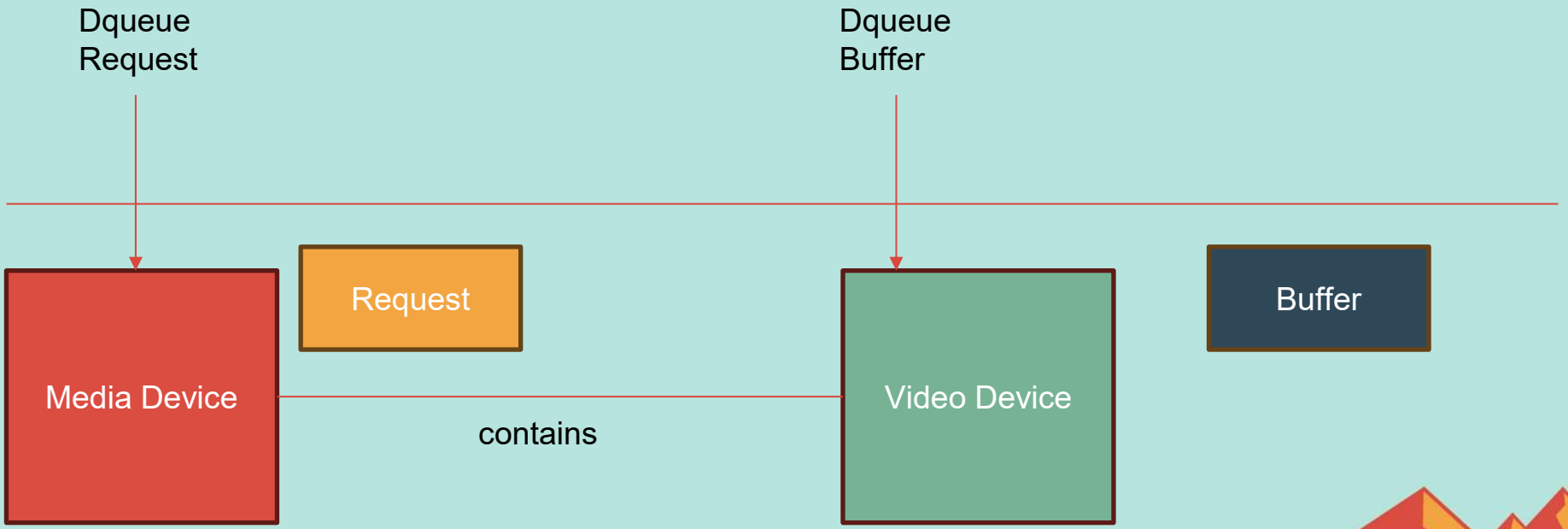
Request API Basics



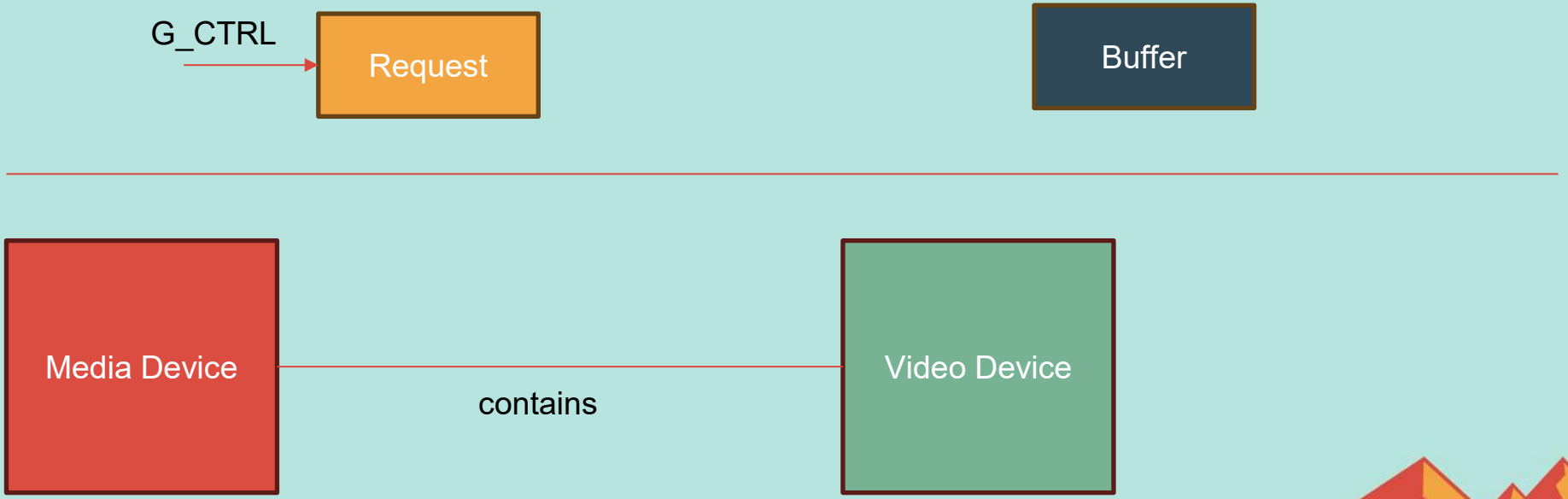
Request API Basics



Request API Basics



Request API Basics



Request API Basics

```
struct v4l2_buffer buf;
struct v4l2_ext_controls ctrls;
int req_fd;
...
if (ioctl(media_fd, MEDIA_IOC_REQUEST_ALLOC, &req_fd))
        return errno;
...
ctrls.which = V4L2_CTRL_WHICH_REQUEST_VAL;
ctrls.request_fd = req_fd;
if (ioctl(codec_fd, VIDIOC_S_EXT_CTRLS, &ctrls))
        return errno;
...
buf.type = V4L2_BUF_TYPE_VIDEO_OUTPUT;
buf.flags |= V4L2_BUF_FLAG_REQUEST_FD;
buf.request_fd = req_fd;
if (ioctl(codec_fd, VIDIOC_QBUF, &buf))
        return errno;
```

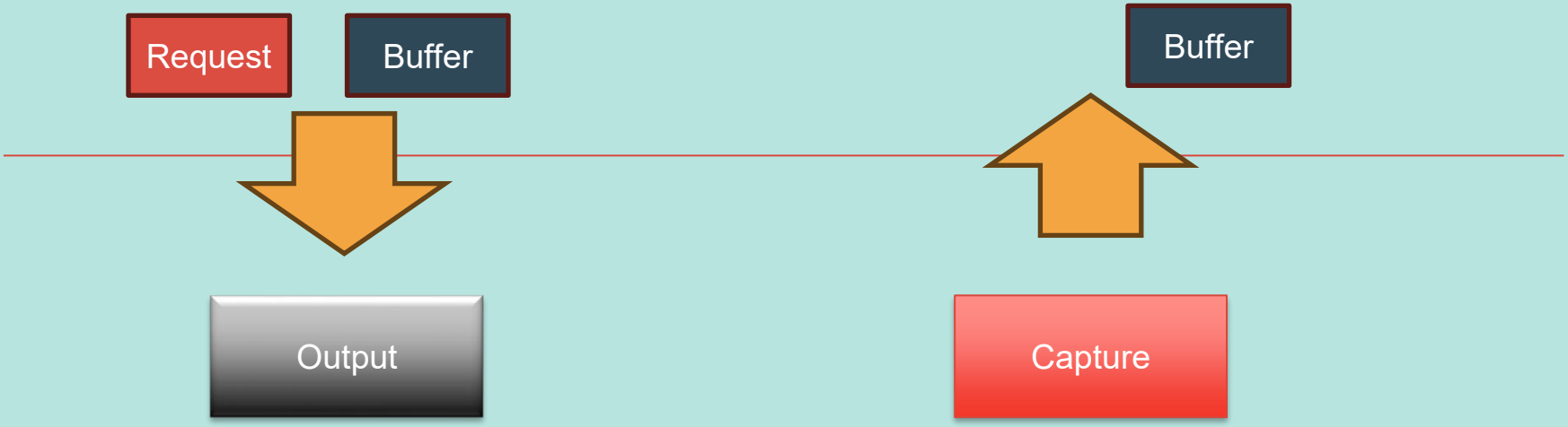
Request API Basics

```
if (ioctl(req_fd, MEDIA_REQUEST_IOC_QUEUE))
        return errno;
```

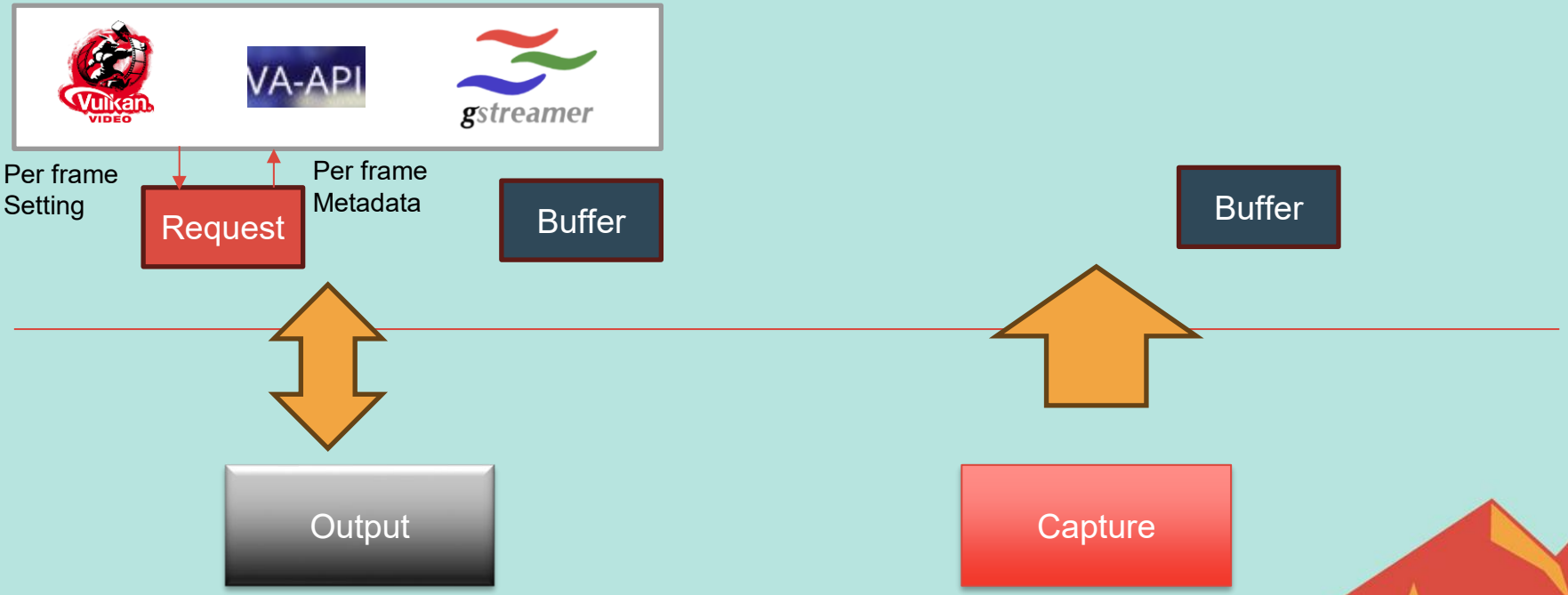
```
struct v4l2_buffer buf;

memset(&buf, 0, sizeof(buf));
buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
if (ioctl(codec_fd, VIDIOC_DQBUF, &buf))
        return errno;
```

```
struct pollfd pfd = { .events = POLLPRI, .fd = req_fd };
poll(&pfd, 1, -1);
...
ctrls.which = V4L2_CTRL_WHICH_REQUEST_VAL;
ctrls.request_fd = req_fd;
if (ioctl(codec_fd, VIDIOC_G_EXT_CTRLS, &ctrls))
        return errno;
```

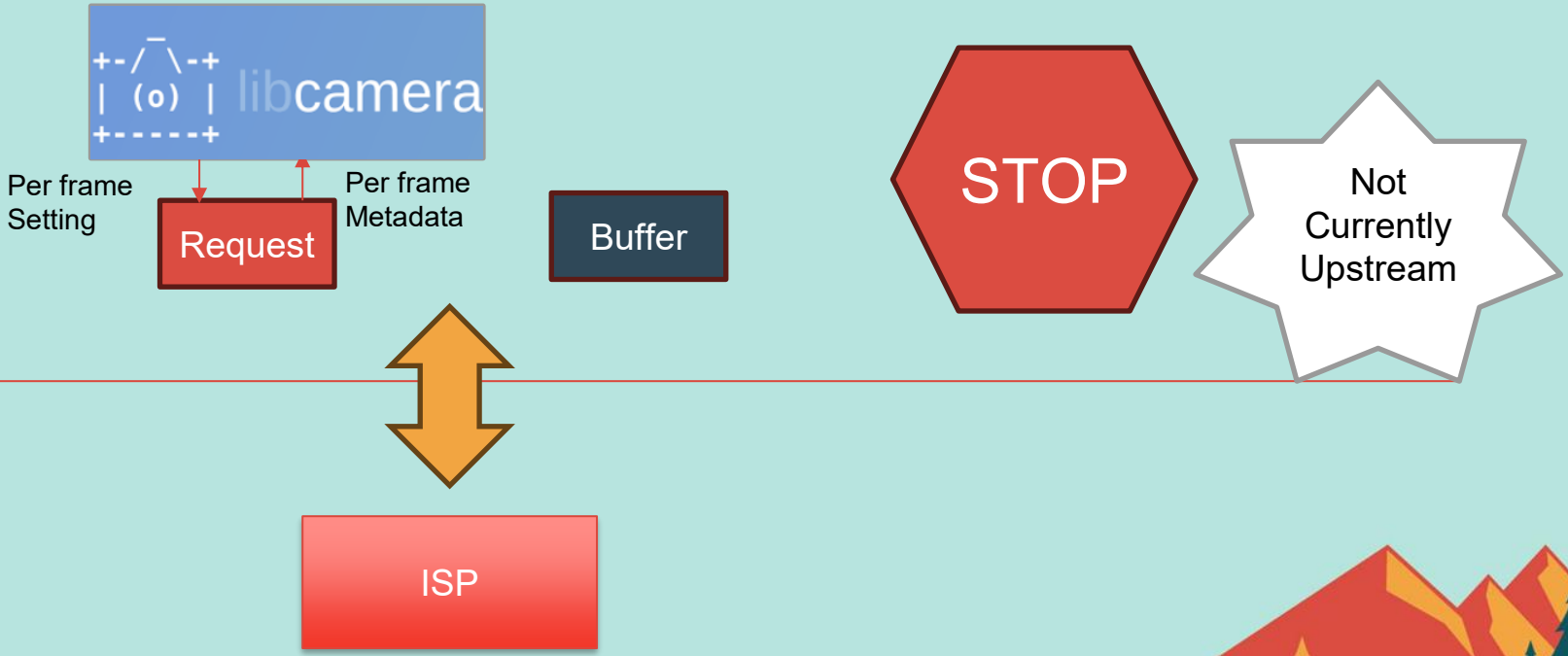
Generic V4L2 M2M Example



V4L2 Stateless Decoder



Complex Camera/ISP



References

- https://docs.kernel.org/userspace-api/media/mediactl/request-api.html
- https://docs.kernel.org/userspace-api/media/v4l/dev-stateless-decoder.html
- https://docs.kernel.org/driver-api/media/v4l2-controls.html#c.v4l2_ctrl_request_setup
- https://docs.kernel.org/driver-api/media/v4l2-controls.html#c.v4l2_ctrl_request_complete

Questions ?