V4L2: multiple streams

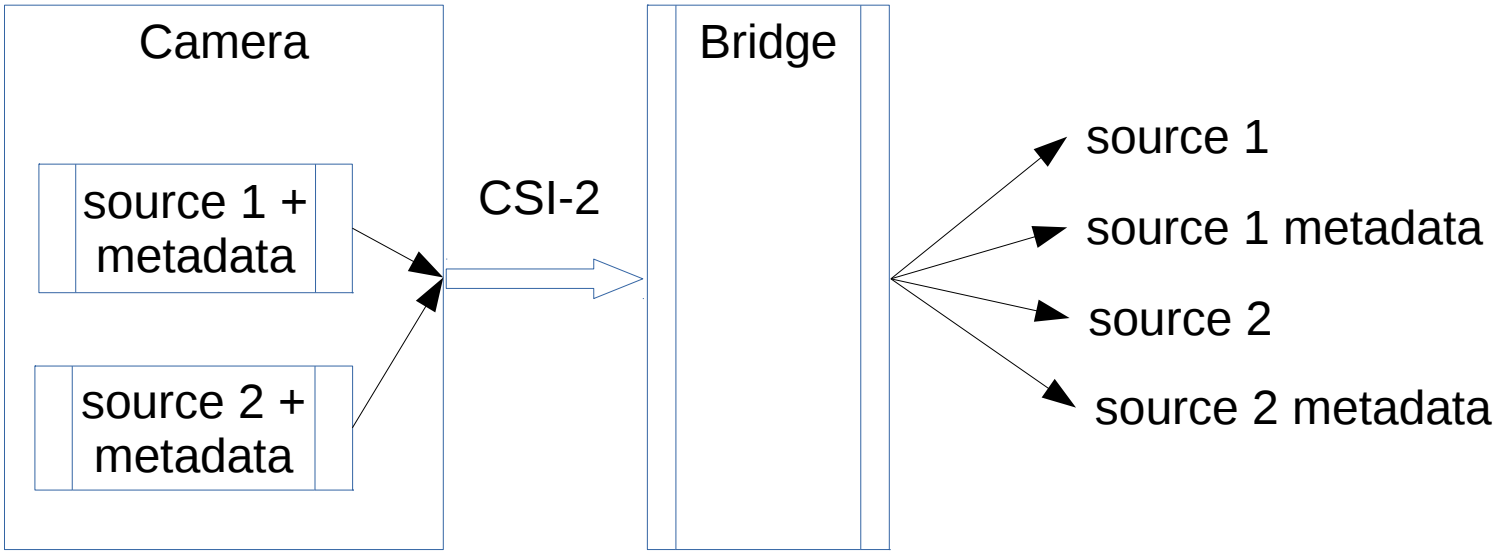
Guennadi Liakhovetski
<g.liakhovetski@gmx.de>

Background: CSI-2 data multiplexing

- Data on MIPI CSI-2 (and CSI-3) buses is transferred in frames.
- Each frame has a header, which contains a byte, describing the “kind” of data, contained in the frame.
- 2 bits in that byte define a Virtual Channel. A CSI-2 device can send data on up to 4 virtual channels.
- 6 bits define a Data Type. Many data types are defined by the standard. Some are reserved. Several codes are used for user-defined data-types.

Goal design

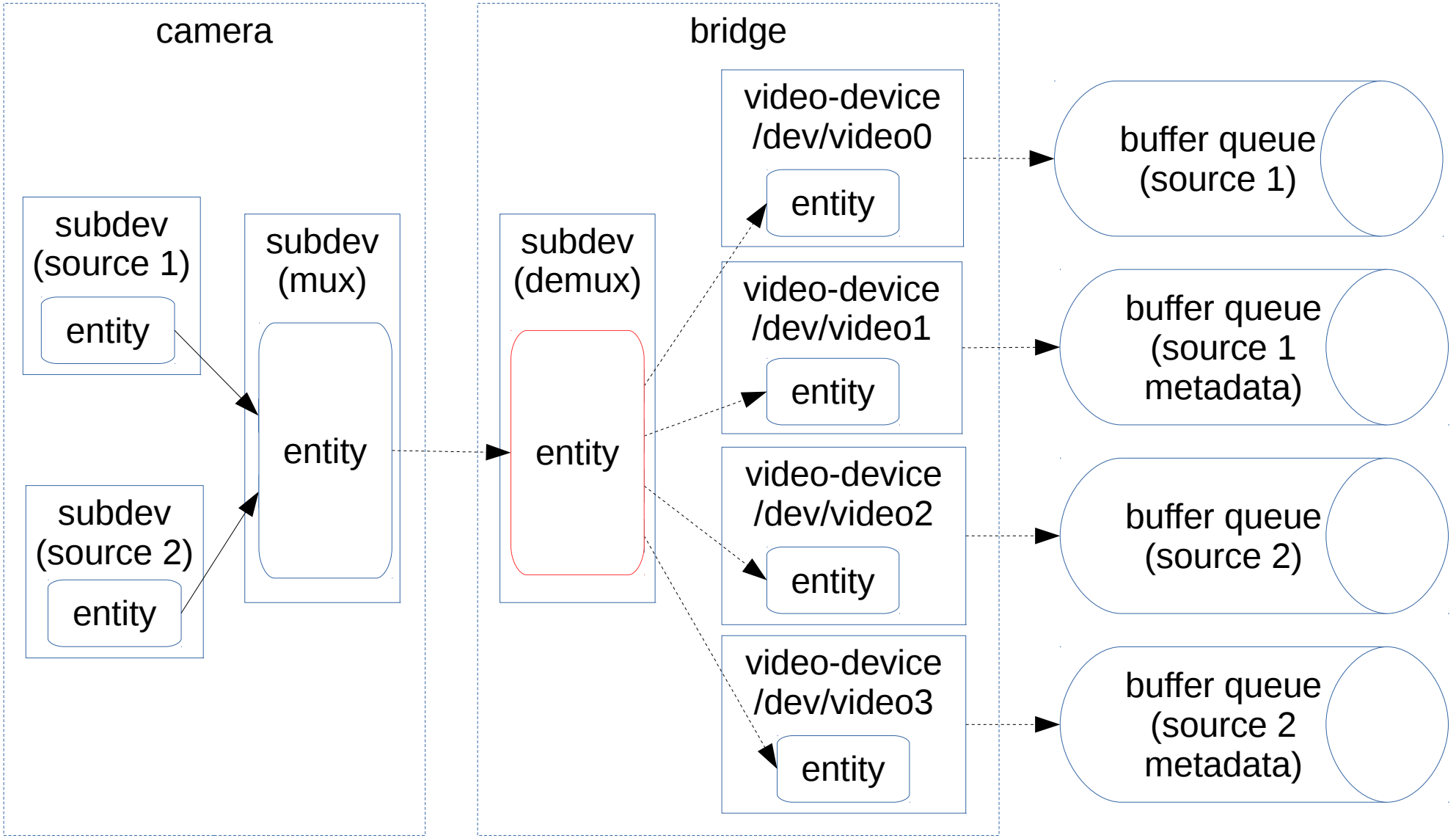
- The goal is to control and capture multiple streams independently.
- Disregarding technical design, an example representation can look as follows, assuming, that each source has associated metadata:



Example: 3D camera

- 2 sensors: depth and image.
- Sensors use different CSI-2 virtual channels.
- Sensors send image data and metadata, using different CSI-2 data types.
- Sensors must be coupled for calibration and synchronisation.

Proposed implementation



Discussion: camera

- Some sensors may stream multiple image data types simultaneously, e.g. Bayer and JPEG. In such cases additional subdevices have to be implemented, e.g. a JPEG compressor.
- The internal camera topology is hardware-specific.
- If metadata is not configurable, its explicit configuration within subdevices can be omitted.
- Each sink pad on the multiplexer subdevice is configured for one data type and virtual channel.

Discussion: bridge

- Each data type on each virtual channel has to be made available on a separate video device node.
- The number of video nodes corresponds to the maximum number of streams, that the bridge can support. It can be large, especially if there are multiple bridge instances on the system.

Discussion: mux / demux

- The link between the multiplexer and the demultiplexer is configured with a fixed format.
- An API can be implemented for configuration and validation of such multiplexed links.

V4L2 modifications

- Video devices will link to a demultiplexer subdevice. We will configure those links for specific streams, that we want to capture, by setting a virtual channel number on respective demultiplexer pads. This is the only V4L2 extension, that we need. The respective location in the chart is marked red.
- Optionally an explicit V4L2 mux/demux API could be implemented for configuration and validation of multiplexed links.
- Multiple buffer queues per video device could be considered as well, which would eliminate large numbers of video nodes.

Discussion: processing unit

- Some processing units can process multiple streams from multiple sources: