Media Controller for DVB

Linux Media Summit

Mauro Carvalho Chehab
Linux Kernel Expert
Samsung Open Source Group

Mar 26, 2015
There are several device nodes for Digital TV to control hardware components:

- `/dev/dvb/adapter/?/frontend?` - controls the tuner, demod and SEC
- `/dev/dvb/adapter/?/ca?` - controls the conditional access module;
- `/dev/dvb/adapter/?/demux?` - controls the demux

There are other device nodes:

- `/dev/dvb/adapter/?/dvr?` - for the MPEG-TS filtered output
- `/dev/dvb/adapter/?/net?` - controls the MPEG-TS filter for a network adapter
Digital TV data flow pipeline

- Via DVB demux interface, filters are dynamically created/removed
  - Each filter contains a PID (PES filter) or a section filter (to filter tables)
- A PID set is output to userspace via a dvr devnode
  - Eventually after passing through CAM
  - Each single PID could, instead be sent to a per/PID file descriptor on demux devnode
- On embedded hardware, the sink can actually be a GPU pipeline.
Digital TV control pipelines

- Several device nodes are used to control the hardware
  - There's currently a discussion about how to represent the control devnodes
    - As a property to the block?
    - As control entities?
  - multiple devnodes may control different aspects of the same device block
    - net? and demux? devnodes, for example controls the same demux
  - dvr? device nodes don't control anything. They're used just for data I/O
Digital TV without DMA data flow

On embedded hardware, the sink can actually be a GPU and audio pipelines:

No data is transferred to the OS; all data flow happens inside the hardware.

This is actually a requirement on some DRM-protected hardware.
 Via DVB **net** interface/devnode, filters and network interfaces are dynamically created/removed

- Each filter contains:
  - a single PID with contains IP traffic
  - encoding: Ultra Lightweight Encoding (ULE) or Multi Protocol Encapsulation (MPE)

- The *dvb* network interfaces contain ethernet-like frames
  - with TCP/IP stack inside it, and a Maximum Transfer Unit (MTU) equal to 4096 bytes
  - The interfaces are dynamically created/removed when the filter is set/deleted
Embedded Set Top Box hardware

Antenna

Diplexer

Low Noise Amplifier

Splitter

Tuner

To QAM + Analog Demodulator for Picture-in-Picture

Tuner

To QAM + Analog Demodulator for Main Video

Tuner

To QAM + Analog Demodulator for Video Recorder

DOCSIS 1.1 Modem

From QPSK Modulator for Cable Modem

Upstream Amplifier

From QPSK Modulador For Cable Modem

Such hw has up to:
- 4 Tuners
- 3 analog demods
- 3 digital demods
- 1 modulator

Putting altogether (data flow only)

PS.: Fictional graph, with cross-bars for everything after tuner, based on the previous graph
Issues to be addressed

- How to represent control dataflow?
- How to represent hw control-only entities that don't have data flow associated with, like SEC, flash, etc?
- Nomenclature for device nodes
- Nomenclature for DMA and kernelspace-userspace transfer units
- How to handle software-only processing blocks
- Dynamic creation/deletion of entities and pads (needed for DVB network virtual devices)

References:
- https://www.mail-archive.com/linux-media@vger.kernel.org/msg85910.html
- https://www.mail-archive.com/linux-media@vger.kernel.org/msg85979.html
Other issues with MC

1) dynamic creation/removal of pipelines

2) media_entity_pipeline_start
   - It should be possible to pass not only the start entity but also the final entity, as it should be possible to control tuner/fe pipeline independently of the demux filter pipelines.

3) pipelines with audio and DRM

4) possible race contition between activating the pipeline via media_entity_pipeline_start() and marking it as busy (to be checked)

• References
  - https://www.mail-archive.com/linux-media@vger.kernel.org/msg83883.html
  - https://www.mail-archive.com/linux-media@vger.kernel.org/msg83884.html