Testing the Media Subsystem: Compliance Tests and Virtual Drivers

Hans Verkuil
Cisco Systems Norway
hverkuil@xs4all.nl

History, Features & Architecture

History

- First drivers appeared in 1996: bttv and bw-qcam.
- V4L1 API appeared around 1999.
- V4L2 API appeared around 2002.
- V4L1 API was removed end 2010/beginning 2011.
- Support for complex video pipelines was finalized end of 2012.
- Close relationship with DVB and IR. V4L2 + DVB + IR are all part of drivers/media.
- The media HDMI CEC subsystem was added in 2016.

Features of V4L2

- Video capture/output and tuning (/dev/videoX, streaming and control)
- Raw and Sliced VBI capture and output (/dev/vbiX, streaming and control)
- Memory-to-Memory (aka codec) devices (/dev/videoX, streaming and control)
- Radio tuning and modulating (/dev/radioX, control, ALSA for streaming)
- Software Defined Radio tuning and modulating (/dev/swradioX, control, ALSA for streaming)
- RDS receiver/transmitter (/dev/radioX, streaming and control)
- Device topology discovery/control (/dev/mediaX, control)
- Low-level sub-device control (/dev/v4l-subdevX, control)
- Touch devices (/dev/v4l-touchX, streaming and control)

Video HW Architecture

- Constellation of devices:
  - DMA engine
  - Sensor
  - Video receiver
  - Video transmitter
  - Tuner
  - Demodulators
  - IR receivers/transmitters
  - Muxers
  - Audio amplifiers
  - Audio mixers
  - De-ghosting
  - Comb filters
  - Scalers
  - Image processors
  - Camera flash
  - RDS encoders/decoders
  - Modulators
  - Codecs

Driver architecture

- The bridge driver controls the platform/USB/PCI/... hardware that is responsible for the DMA transfers.
- Based on the board configuration (USB ID, PCI ID, kernel config, device tree, module options) the necessary *sub-device* drivers are loaded.
- The bridge driver finally registers the device nodes it needs.

Lots of IOCTLs

- V4L2 API (videodev2.h): 82 ioctls
- V4L2 subdevice API (v4l2-subdev.h): 25
- Media Controller API (media.h): 8

IOCTLs per Category

- V4L2 Core (capabilities, controls, events, debug): 20
- Input (enumerating, selecting): 6
- Output (enumerating, selecting): 6
- Tuning/Modulating: 8
- Analog TV: 8
- Digital Video (HDMI/DisplayPort/etc): 14
- Format + Cropping/Composing: 24
- Streaming: 9
- Codecs: 7
- Routing: 6
- Miscellaneous: 11

How To Test?

Problems

- A vast variety of hardware with wildly different feature sets, often (very) difficult to obtain.
- A large API to test, you would need a lot of hardware to be able to cover all those features.
- Media subsystem maintainer: how to check for regressions in the media core frameworks?
- Driver developer: how to test my driver and how to check if my driver is implementing the API correctly?
- Application developer: how to test if my application can handle all the different types of hardware?
- In some cases: how to test that a remote device connected to our device is compliant? (HDMI CEC)

Solutions

- Create a compliance test utility that driver developers can run against their driver to verify compliance. When submitting a new driver (or making major driver changes) the compliance output has to be included in the cover letter.
- Not perfect, but helps enormously when doing code review (it catches all the silly corner cases), and gives a lot more confidence in the driver.
- Testing core framework changes is pretty much impossible with real hardware, instead we created virtual drivers (drivers emulating hardware). These drivers create devices with the widest possible feature set.
- Core framework changes can be tested against those drivers using the compliance tests: these should still pass.
- Application developers can test against those drivers, no need to buy a lot of hardware for testing.
- Emulating hardware allows for error injection to test corner cases.

v4l2-compliance

- Started 15 years ago, all it tested where a handful of core ioctls.
- It took 6 years to finally get the tests for streaming in, and another year to test formats and crop/compose combinations.
- New drivers must pass the compliance tests.
- Close to 1000 tests are performed.
- Tests for new APIs must also be added to the compliance test. Great way to verify if proposed API is sane.
- It is more strict than the V4L2 specification: it assumes drivers use the correct core frameworks which means that there is no excuse to e.g. support VIDIOC_S_CTRL but not VIDIOC_S_EXT_CTRLS.
- Always compile from the v4l-utils git repository to get the latest tests.
- Tests are simple: `fail_on_test(controls != num_regular_ctrls);`

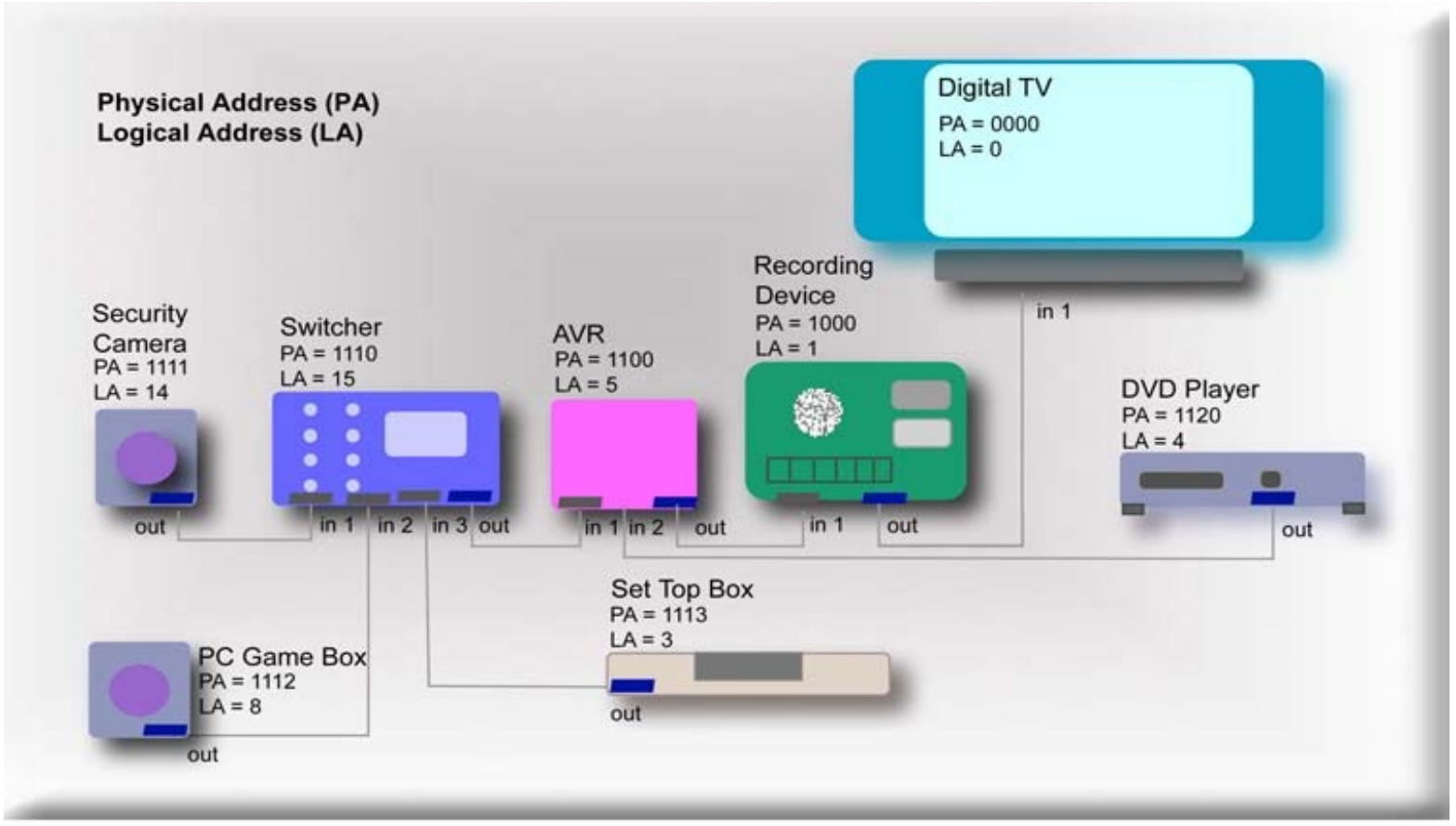
```
fail: v4l2-test-controls.cpp(356): controls != num_regular_ctrls
```

Test Drivers

- vivid: video capture & output, vbi capture & output, radio receiver & transmitter, software defined radio capture, metadata capture & output, touch capture, HDMI CEC emulation. Closely emulates what 'real' hardware will do. Emulates a webcam, analog SDTV video, SDTV TV tuner and HDMI digital video capture.
- vim2m: memory-to-memory video scaler test driver.
- vicodec: memory-to-memory video codec test driver.
- vimc: Camera ISP-like test driver.
- visl: test driver to test stateless codec APIs.
- Most V4L2 devices only support a (very) limited subset of the V4L2 API. Without test drivers (or a huge collection of hardware) it is impossible to test your application, but with these drivers you can.

Demo!

HDMI CEC:
Consumer Electronics Control



CEC @ 1 meter: 400 bits/s

Testing CEC

- cec-compliance: can test your device, but also a remote device whether it follows the CEC protocol.
- The vivid test driver can emulate CEC, useful for regression tests and application testing.
- The cec-gpio driver can drive the CEC pin when connected to a GPIO. This drives the pin directly and allows low-level error injections to test all sorts of error conditions or rare situations (e.g. Arbitration Lost).

Demo!

Resources

Resources

- Linux Media Infrastructure API: http://linuxtv.org/docs.php.
- Media subsystem repository: https://git.linuxtv.org/media_tree.git
- v4l-utils git repository: http://git.linuxtv.org/v4l-utils.git
- linux-media mailinglist & irc channel: http://linuxtv.org/lists.php

Questions?