Digital TV Kernel Pipelines via Media Controller API

Mauro Carvalho Chehab
Linux Kernel Expert
Samsung Open Source Group

Mar 25, 2015
Digital TV devices

- A Digital TV device consists on a set of hardware blocks. The basic components are:
  - **Tuner:**
    - Tune into a physical frequency (tuner), and output the channel on an intermediate frequency (IF);
  - **Demodulator (a. k. a. demod):**
    - Gets an IF, decodes the sub-carrier(s) content and outputs the resulting MPEG-TS stream. It is specific for a given set of DTV standards;
  - **Demultiplexer (a. k. a. demux):**
    - Filters the MPEG-TS, extracting video, audio and other data information, like subtitles, Electronic Program Guide, etc
    - The demux may also extract TCP/IP packets from the MPEG-TS and send them via Linux network interfaces

**NOTES:**
- Satellite devices also have a Satellite Equipment Control (SEC), with controls external components at the antenna subsystem (switches, LNBf, rotors, …)
- Cheap devices don't have demux. Linux Kernel emulates it on such cases.
- Some devices may have a MPEG-TS multiplexer (for TCP/IP data send) and Conditional Access Module support (for Digital Rights Management).
Media devices block diagram

PS.: picture is not complete: other blocks may be present
Digital TV frontend

- The DTV frontend is the hardware part that:
  - Tunes into a physical channel;
  - Demodulates the channel data;
  - Controls the satellite and signal amplifiers.
- So, it consists of several sub-devices: tuner, demod, amplifiers and SEC.
- On Digital TV, tuning into a channel is a tightly coupled operation:
  - The IF used by the demod and tuner should be the same;
  - The tuner filters should be optimized to the digital TV standard in usage;
  - On some devices, the demod should control the tuner/amplifiers gains and set bandwidth filters dynamically, in order to increase the quality of the signal;
  - On Satellite devices, sometimes the same hardware component have internally all 3 functions;
  - On several devices, a FPGA or a micro-controller handles the both tuner and demod. So, the tuner is not directly visible;
- The Linux DVB API was designed to expose the frontend as a single entity.
  - Yet, we may need to expose the sub-devices in some future
MPEG-TS container example

Virtual Channel 17.1: playing movie (HD)
- Video PID 100
- Audio 1 PID 101
- Audio 2 PID 102

Program 1 (service ID 1000)
Program 2 (service ID 2000)
- Video PID 200
- Audio 1 PID 231
- Data PID 554

Virtual Channel 17.2: playing news (SD)

NOTE:
The Packet ID (PID) and Service ID data are described by some tables inside the MPEG-TS: PAT, PMT plus: SDT(DVB, ISDB) or TVCT/CVCT (ATSC)
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 though 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.
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.
Network (MAC) pipelines

- 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

Antena

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

Tuner

Upstream Amplifier

From QPSK Modulador For Cable Modem

Such hw has up to:

4 Tuners
3 analog demods
3 digital demods
1 modulator

Media Controller (MC) API

- Designed originally for V4L2 devices
- Shows/changes the device's pipelines
- Focused on embedded devices
- Subdev API: controls each logical element on complex devices
  - Not sure yet if this is needed for DVB
Media controller API

- Discovering a device internal topology
  - hardware devices are modeled as an oriented graph
  - building blocks called entities connected through pads.

- An entity is a media hardware or software building block
  - correspond to logical blocks: physical/logical hardware devices, DMA channels, physical connectors

- A pad is a connection endpoint
  - Represents interactions between entities
  - Data flows from the entity's output to one or more entity inputs

- A link is a point-to-point oriented connection between two pads
  - Data flows from source to sink pads
Example 1: A Siano Rio ISDB-T digital USB stick

$ media-ctl -p
Media controller API version 0.1.0

Media device information
------------------------
driver                usb
model                 Siano Rio Digital Receiver
serial
bus info              1
hw revision           0x8
driver version        3.19.0
Digital TV mapping via MC (2)

Device topology
- entity 1: dvb-demux (2 pads, 2 links)
  type Node subtype DVB DEMUX flags 0
  device node name /dev/dvb/adapter0/demux0

pad0: Sink
<- "Siano Mobile Digital MDTV Recei" : 1 [ENABLED]
pad1: Source
-> "dvb-dvr" : 0 [ENABLED]

- entity 2: dvb-dvr (1 pad, 1 link)
  type Node subtype DVB DVR flags 0
  device node name /dev/dvb/adapter0/dvr0

pad0: Sink
<- "dvb-demux" : 1 [ENABLED]

- entity 3: Siano Mobile Digital MDTV Recei (2 pads, 1 link)
  type Node subtype DVB FE flags 0
  device node name /dev/dvb/adapter0/frontend0

pad0: Sink
pad1: Source
-> "dvb-demux" : 0 [ENABLED]
Digital TV mapping via MC (3)

Siano Mobile Digital MDTV Recei
/dev/dvb/adapter0/frontend0

dvb-demux
/dev/dvb/adapter0/demux0

dvb-dvr
/dev/dvb/adapter0/dvr0

Note that the tuner subdev is not shown.

In this specific device, the tuner is not even visible to the driver, as a micro-controller handles it internally.

Created with: media-ctl --print-dot|dot -T svg -o siano.svg
Digital TV mapping via MC (4)

Example 2: An hybrid analog TV/ISDB-T device based on cx231xx
Current status

- Currently, experimental patches were merged at linux-media development tree, with initial Media Controller support for DVB

- Several drivers already exposing the DVB device nodes via the Media Controller API:
  - Siano sms1xxx driver;
  - Conexant cx231xx driver – helps to demonstrate the hybrid analog/digital case;

- Only the device nodes are currently created
  - And tuner subdev, for hybrid devices

- The dynamic per-filter part of the pipeline and the demux sink is not represented

- No subdev API usage for DVB yet

- Discussions will happen at the Linux Media Summit, on March, 26, in order to address the pending stuff.

You're invited to join us
How to contribute

- Main discussions and patches for TV on Linux:
  - Userspace/kernelspace: linux-media@vger.kernel.org

- Upstream trees:
  - To test Kernel drivers: http://git.linuxtv.org/media_build.git
  - To develop Kernel drivers: http://git.linuxtv.org/media_tree.git
  - V4l-utils, including media-ctl: http://git.linuxtv.org/v4l-utils.git

- Documentation:
  - Media APIs http://linuxtv.org/downloads/v4l-dvb-apis

- Wiki pages: http://linuxtv.org/wiki/

- IRC channel: irc.freenode.net
  - Digital TV channel #linuxtv
  - Can be assessed via http://webchat.freenode.net/
Thank you.

Questions?