Adding DVB TV and Radio to AOSP TV

Bernhard “Bero” Rosenkränzer <bero@linaro.org>
Why? Isn’t streaming over the net the future?

- DVB still offers numerous free channels (The Astra TV satellite, available all over Europe, transmits over 800 free-to-air channels). It is unlikely to go away for that reason alone.
- DVB-T/DVB-T2 is the official transmission standard of public TV and Radio channels in many countries
- DVB-S, DVB-T and sometimes DVB-C are usually available even in regions where Internet access is still too slow or too expensive for streaming video.
How?

- AOSP/Android hasn’t standardized on a way to address DVB receivers.
- In the regular Linux world, the (more or less) agreed upon standard is using libdvbvt from v4l-utils…
- … which would technically work on AOSP as well, but it’s licensed under the GPLv3, so while it would work for us, it would never be accepted by upstream AOSP
- Since upstream acceptance is a goal, we had to ignore libdvbvt and start from the kernel DVB interface.
So maybe we can do better?

- Some things in the DVB standard pretty much call for generic code - lots of DVB tables and descriptors to be read/parsed, all following the same general idea, but with different data, different order, …
- Coming up with a readable API for this pretty much calls for a C++ (or similar) library that can make use of inheritance and a few templates
- Given it isn’t just a replacement for libdvbv5 anymore, would be nice to be able to run it on normal Linux as well…
- Yet, it may be useful to optionally make use of AOSP-isms (get DVB streams through binder in a binderized HAL instead of reading the device file directly or streaming over a Unix or TCP socket)
What can the API look like?

Sample code scanning for channels:

```cpp
dvbiinterfaces cards = dvbiinterfaces::all();
for(DvbiInterface &c: cards) {
    c.tune(initial);
    std::vector<Transponder*> tp = c.scanTransponders();
    for(auto const &t: tp) {
        if(c.tune(t)) {
            std::vector<Service> srv = c.scanTransponder();
            for(auto const &s: srv)
                std::cout << s.name() << std::endl;
        }
    }
}
```
What can the API look like?

Sample code setting up the interface for receiving a channel:

```cpp
auto dev = DVBInterfaces::all()[0];
ChannelList channels("channels.dvb");
auto channel = channels.find(argv[1]);
dev.tune(channel.first);
dev.setup(*channel.second);
```

That’s all -- /dev/dvb/adapter0/dvr0 now outputs the MPEG transport stream of the requested channel
Can we do even more?

Yes -- for example, we can get data from multiple channels on the same transponder at the same time (e.g. watch one, record 2 others). The API for this isn’t finalized yet, but sample code exists inside the source tree.

https://github.com/berolinaro/dvbv5-tools

It runs on both AOSP and regular Linux. A fairly modern toolchain (clang >= 4.0 or gcc >= 6.0) is required.
Can we do even more?

Another code sample in the tree grabs data from a DVB card and retransmits it over http -- Let’s try to watch some European TV by talking to my setup at home…

The data is currently sent by my desktop box with a PCIe DVB-S receiver for stability (nobody home to kick the box if something breaks before Demo Friday…), but I’ve had the same setup working on a 1GB Hikey with a USB DVB-T receiver stick before.
Next step: Integrating into the AOSP TV UI

Currently the basic functionality is running, and supports all 5 major DVB variants (DVB-S, DVB-S2, DVB-C, DVB-T, DVB-T2). Adding ATSC support should be easy.

Integration into the AOSP TV user interface isn’t finished yet (but I’ve started looking at the interfaces), but we know it can work.

Possible challenge: ExoPlayer doesn’t support MPEG Transport Streams (mpv, vlc 3.x and ffplay do) - we may have to convert the Transport Stream into a Program Stream before feeding it to AOSP TV. But such converters exist (but are GPL - so for the sake of upstreaming may need to be rewritten).
Questions or comments?

Feature requests?
Thank You

bero@linaro.org

#SFO17

SFO17 keynotes and videos on: connect.linaro.org

For further information: www.linaro.org