The Architecture of the Open-Source Remote Phy Device

Alon Bernstein
Anlu Yan
Cisco Systems

Abstract

Cablelabs has recently announced the open-sourcing of the remote phy device (RPD). This paper outlines the vision and main components of the open software that will drive the RPD. For those who seek an in-depth review it's recommended to sign up to cablelabs openRPD project, read the documentation and of course read the source code !

MOTIVATION AND VISION

Open source has evolved over the years, from a lets-all-be-friends culture of code sharing to an economically driven joint development framework. The original 1985 GNU manifesto stated that "The fundamental act of friendship among programmers is the sharing of programs," and that the commercialization of software is bad. For years to follow the mental image of an open source contributor was that of a poorly groomed collage kid hacking away code late at night. While there are elements of this visual that are true, it turns out that open source is a big business that employs many salaried adults. "Open" should not be confused with "free" and as we explain in this paper at the base of it open source is a joint development framework that can help reduce costs, accelerate feature development and increase software quality for the following reasons:

1. Cablelabs members have jointly developed specifications ever since cablelabs was created. This has proved a very successful model for both operators and vendors. Open source represents a natural next step for this existing collaboration model since published code can be viewed as a super-detailed definition of how a feature should work.
2. As the industry has experienced in the past, publishing an interface specification does not guarantee smooth sailing for inter-operability tests. Open source has great potential in shortening the interop stage because all vendors share the same interface code base.
3. Open source does not mean that there is no vendor differentiation. On the contrary: the open source components should focus on the functions that provide no vendor differentiation, for example parsing messages, reading various sensors, writing alarms to logs etc, thereby allowing a vendor to focus on those features and components that provide added value.
4. The parts of the code that are open source get tested and deployed in a large number of systems, not only a single vendor system. Because of the licensing agreement any fix to the open source part of the code gets committed back to the project resulting in higher software quality overall.
5. With open source the operator community and vendor community have an opportunity to closely collaborate, this is especially relevant as many of the larger operators are large software development houses as well.
6. "open source" is not only about source code, it's a complete SW development environment including source control, bug tracking and simulation code. All of the above is supported by Cablelabs.

7. And last but not least, open source is good for the engineering staff. Their open source contribution are the ultimate referral tool, and the communities that they build is a safety net and support system.

LICESNING AND PRODUCTIZATION

Open source projects are not products. There is no commercial support structure around them. Open source is a base on which products can be built. For example, the open source code from linux.org is not used directly. There are several commercial distributions (e.g. redhat, android) that use the linux kernel and provides support and enhancements around it.

The open source licensing is put in place to make sure there is reciprocation in the community that the software can be used but bug fixes and feature enhancements are provided back to the code base. There are a couple of licenses that are typically used in open source projects, most notably GPL and Apache. The Apache license is considered a very permissive type of license.

From the licencing point of view, the OpenRPD project consists of two parts. One is the OpenRPD packages and the other is enhancements and modifications to OpenWrt. The OpenRPD packages consist of Remote PHY specific contents, to run as user level processes and communicates with the OpenWrt part via IPCs. This part is licensed under Apache 2. The OpenWrt part consists of Linux kernel and other user level processes, some with OpenRPD specific modifications and enhancements. This part retains the licenses of the original packages, many of which are GPL.

KEY ARCHITECTURAL CHOICES

The following sections outline the main tenants of the open RPD software and the reasons for eastablishing them.

This document does not discuss the hardware design of the RPD, however as a reference it's useful to present an overview of the hardware. `Figure 1` depicts a high level diagram of the main hardware components in the RPD design.

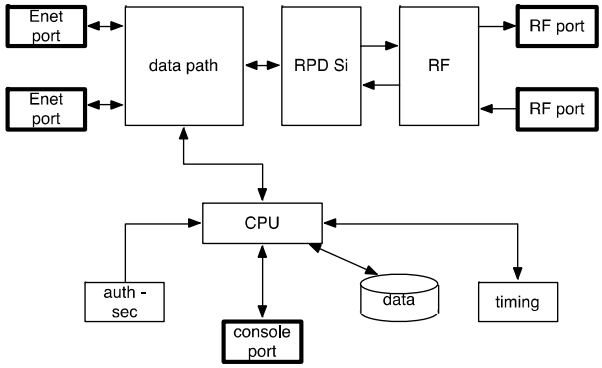

Figure 1 RPD hardware overview

For the most part the vendor specific portions of the software are to be the drivers for the data path, the RPD Silicon and the RF interface. As noted later there is a common HAL (hardware abstraction layer) that provides a uniform, yet extensible, interface to program and read stats from the hardware. Many of the software modules above the HAL are generic and provide little differentiation as they focus on the standard registration/authorization/authentication/etc. processes that have been already standardized by Cablelabs.

OPERATING SYSTEM

The base OS for the openRPD is openWRT, an embedded operating system based on Linux kernel. While openWRT was originally designed for home gateways it is a good base for the openRPD because its designed for embedded systems and has and active community around it and a rich library (more then 3500 packages) that guarantees its longevity as a platform and the ability to add capabilities into it.

MODULARITY, MODULARITY, MODULARITY

The key to a successful project, and an open source project specifically, is a modular design. This is critical because of several reasons:

1. For openRPD vendors developing a product in parallel to the open source system. Certain modules can be implemented in a proprietary way because of time to market pressure or other reasons. If the base infrastructure is modular it's easy to swap these components if and when an open source equivalent exists.
2. The same argument applies in the reverse as well. A vendor may start with an open source module and later replace it with a proprietary one.

The main modules for the open RPD are depicted in `Figure 2` and for the most part correspond to the Remote phy architecture as defined by cablelabs in the remote phy specitication specification (http://www.cablelabs.com/specification/remote-phy-specification/):

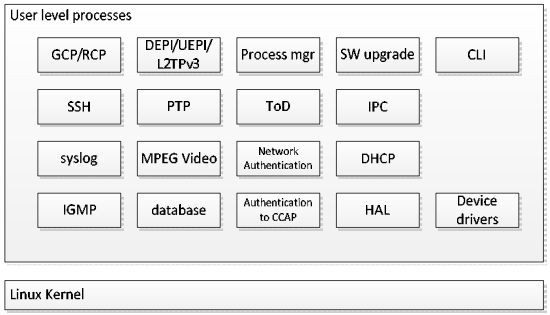

Figure 2 Main software modules for open RPD

To assure modularity the system components communicate over a bus as depicted in `Figure 3`:

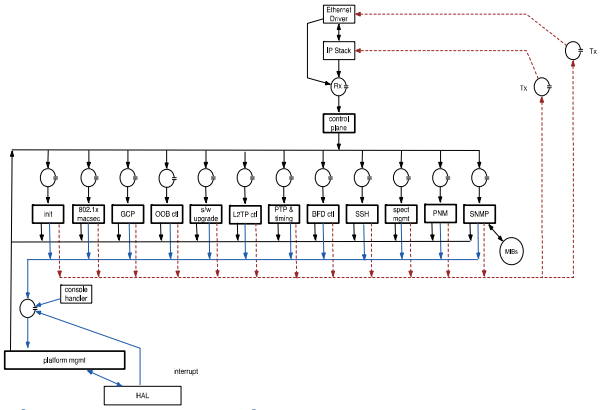

Figure 3 Bus architecture for open source RPD

The IPC technology used for open RPD is ZeroMQ. It's a high-speed asynchronous message library designed for distributed applications and for modularity reasons we treat our components as part of a distributed system even if physically located on the same hardware. ZeroMQ also provides sub/sub and push/pull capabilities for message exchange. With ZeroMQ different modules can be developed in different programming languages and easily swapped into and out of an implementation.

A modular design requires a small overhead because modules communicate over a message infrastracture instead of sending messages directly. However this is relativity insignificant and as Donald Knuth has declared "**premature optimization is the root of all evil**". Clean separation of modules is key to the success for both the RPD open source and RPD products based on it.

GOOGLE PROTOCOL BUFFERS

For internal communications over ZeroMQ the RPD software uses Google protocol buffers (GBP). GBP is a method for serializing data and looks a little bit like JSON definition, mostly used for defining messages in a distributed system.
This choice of GBP further supports the modularity as well since its endian-independent, programming language independent and processor independent

DATABASE

OpenRPD includes a small database (DB) as a generic service which different processes can use in specific ways. One common use is to store persistent data. Persistent data can be used for fast error recovery in cases of process crash. As we can see in the following section the DB plays a critical role in the hardware abstraction layer (HAL) process.

HARDWARE ABSTRACTION LAYER

There is a lot of innovation that can occur at the RPD layer with new technologies coming along such as DOCSIS 3.1 and Full-Duplex DOCSIS. To help accelerate this innovation the open RPD project defines a HAL (hardware abstraction layer) that would make it easy to register multiple drivers from multiple vendors onto the same platform framework. These drivers can be different implementations of existing physical layers or new ones.

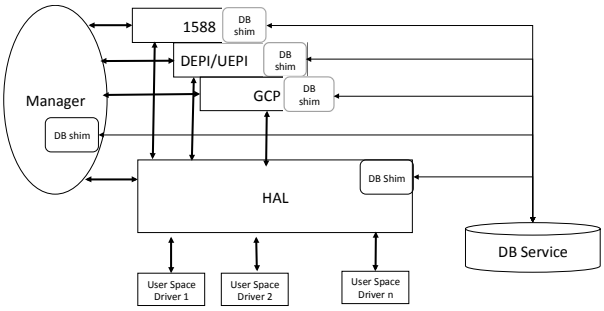

Figure 4 Hardware Abstraction Layer

A key part of the HAL is the database (DB). The HAL uses the DB in the following way:
- GCP may use DB to store shadow data.
- The HAL may use DB to manage data transactions, i.e. GCP stores data in the DB and HAL can pick it from the DB in the following way:

- Each HAL app client (GCP, DEPI, etc.) registers it's supported notification message type to HAL
- Each driver registers its supported message to HAL
- Using a set of notifications the client app knows when the HAL has new data and vice versa.

.

BUILD ENVIROMENT

As open source project is more than a collection of files. It includes a build environment, a bug tracking environment and a test harness. All the above are considered part of the open RPD software.

CONCLUSION

Open source is the next step in vendor and operator collaboration, bringing more consistency and interoperability then the traditional specification writing process and at overall lower cost of implemetation. The open RPD project is hosted and supported by Cablelabs. Readers are encouraged to contact CableLabs and become active contributors to the openRPD project !

REFERNCES

1. openWRT : https://openwrt.org
2. google protocol buffers : https://developers.google.com/protocol-buffers/
3. ZeroMQ : http://zeromq.org
4. Cablelabs remote phy specifications : http://www.cablelabs.com/specification/remote-phy-specification/