Oslec, the Open Source Line Echo Canceller

An Interview with David Rowe

BOB SOLOMON

David Rowe has been working and playing with signal processing hardware and software for 30 years. In 2006 he left an executive position in the satellite communications industry to become a full-time open source developer. Since then David has worked on open hardware and software projects in VOIP, developing world communications, echo cancellation, speech compression, modems, and digital voice over HF radio. David writes a popular blog that is read by 70,000 people each month, drives a home-brew electric car, and also enjoys bike riding and sailing. www.rowetel.com

Phone System Administrator is one of too many hats Robert Solomon wears at a medium-sized nonprofit in New York City. bobsol@gmail.com

Echo is a pain point in open source telephony. The default echo cancellers provided with Asterisk are generally held to be unacceptable [1]. Hardware add-on modules, the common alternative, add $200 or more to the cost of a voice card [2] and are closed source.

In 2008, David Rowe completed development of an improved software echo canceller: Oslec, the Open Source Line Echo Canceller. Although Oslec has been well reviewed by those who have tested it [3], the code has not been incorporated into the Asterisk/DAHDI source. Nine years later, I became aware of Oslec while shopping for new voice hardware. David agreed to an email interview for *;login:*.

Robert Solomon: I am awed by the idea of software echo cancellation for Asterisk that actually works and that a lone developer (with help) could do this. Oslec has at least seven predecessors, none of which were usable IMO. How far along were you before you thought you might succeed?

David Rowe: I always knew it was possible. I mean an EC is just DSP software running on some sort of CPU. However, it took me nearly 20 years of failed attempts, first starting in the early 1990s as part of an early speech codec I was working on. So you might say determination played a part—an unscratched itch—something I had to "fix" at some point in my life.

There were a couple of key algorithms—a way to handle double-talk without diverging and the non-linear echo suppressor. When they dropped into place, I knew I was getting somewhere.

RS: When I first learned of Oslec, surprisingly on a vendor's page offering a hardware EC module [4], and then on your website [5], I thought that you must be really good at maths and algorithms or something. Reading your five-part Oslec blog [6], I learned that the magic of Oslec is more like framing the problem clearly, getting help from people who already know what they are doing, researching the literature, writing the code, and then reaching out to the community for testing. Would you comment? Applied this process in other areas?

DR: Yes, you have the process spot on. A couple of other points are:

1. Developing against the standards-based set of unit tests for EC, which was supported by a framework in Steve Underwood's spandsp library. This neatly isolated any issues and provided a binary pass/fail criteria to develop against.

2. Using open source and offline analysis to "crowd source" testing. Oslec was fitted with test points to capture the signals flowing to the EC. When a beta tester encountered a problem, they could run an application to capture some wave files, then email them to me for offline analysis. This quickly let me engineer solutions to corner cases: for example, low frequency audio from sound cards upsetting the analog hybrids (described in detail in one of the blog posts).

In contrast, everyone else was developing EC by saying "Hello 1,2,3..." down a telephone line in real time.

Oslec, the Open Source Line Echo Canceller: An Interview with David Rowe

I use a very similar process for other signal-processing projects, in particular the use of a high-level simulation that can run "offline" (not in real time) using data captured from the real-time version. For example, in my Codec 2 project, I have GNU Octave simulations that can single step through frames of speech, plotting various signals and statistics. Then, when I'm happy with performance, I run the same code in real time using a bit-exact C port of the same algorithm.

RS: So there is some science involved. "PhD in Electronic Engineering (topic: speech compression)." Want to say anything about that?

DR: Well, the software engineering process was just as important as the science. In addition to signal processing algorithm development, I have a parallel career as a project manager. I, and people working for me, have struggled at times with development of commercial signal-processing widgets. Turns out it's really hard to get the clever maths running effectively in real time in real-world products.

So after having screwed up and licked my wounds a few times, I worked out how to engineer complex signal-processing products effectively. I apply these ideas to my own projects and those I manage for others.

The PhD was on low bit rate speech compression and was completed in the late 1990s. About 10 years later, Bruce Perens approached me—there was a need for an open source low bit rate speech codec. Like echo cancellation, low bit rate speech compression is mired in closed source, license fees, and FUD. So I dusted off the PhD, and the Codec 2 project started [7].

RS: Although Oslec is the default echo canceller in Debian installations [8], echo.ko is not found in any Debian packages except user-mode-linux. I asked Tzafrir Cohen, a Debian DAHDI maintainer, about this in an email and he said that "Debian generally does not ship out-of-tree kernel binary modules." (He recommended building from Debian source with 'm-a'.) I am surprised that a module that was in staging in 2009 is not "in-tree" now. Could you comment, educate, explain?

DR: You know, I've lost track of the progress of Oslec through the staging process. I do recall there was some debate because it was a driver with no hardware. The kernel developer who was managing Oslec in the kernel, Greg KH, may have some comment. It's a good question, and I'd like to know the answer!

RS: Greg KH suggested that I "dig through the public email archives." With the help of Google, I found threads as new as 2012. This prompted me to actually browse the kernel, and I found that echo moved from staging to misc as of 3.15 [9]:

```
2014-02-28      staging: echo: move to drivers/misc/
Greg Kroah-Hartman      6       -0/+1181
```

DR: Yayyy, that means I'm officially a "kernel hacker." :-)

RS: I'm sure there is a good reason why Oslec was not implemented in userspace, but I don't know it. Would you explain?

DR: You need tight control of the delay in speech samples from the ADC/DAC signals flowing from the telephony hardware to the EC. Typically the kernel <-> user mode switch means buffering and timing uncertainty. For the Mesh Potato (village Telco project), I did the EC in user mode, as I built in careful control of the buffering in the kernel mode driver I wrote.

RS: I am noticing a decline of open source telephony in that vendors who once supported "Linux" now, in one case, support only CentOS 6 or worse and, in another case, their special variant of CentOS 7. Would this be due to mobile, to vendor business model, or to the decline of the white box phone system? Some other reason?

DR: I don't feel I have any useful knowledge on this one. I'm not involved in phone systems or Asterisk anymore myself. Given the rise of mobile phones I do wonder about the long-term viability of any sort of PBX; in my day job (a seven-person startup), we don't even have a landline.

RS: Your work on Oslec was complete in 2008 or so. What are you up to these days?

DR: Oslec was developed for the IP04, an open source embedded IP-PBX that I developed. I sold and supported the IP04 for many years in partnership with Atcom, but sales dropped off and, from a technical point of view, I lost interest.

Since 2009 my main project has been Codec 2, a low bit rate (3200 to 700 bits/s) open source speech codec. In the last few years I've been working on the problem of digital voice over HF radio. Making some progress but haven't beat legacy analog Single Sideband (SSB) yet. Along the way, I've developed several high quality modems for digital radio and discovered some more marketing-based FUD—not unlike that around hardware EC. My blog (rowetel.com) is also very popular.

I recently joined an Australian startup (solinnov.com.au) that does contract-based FPGA-based signal processing development for communications and defense applications. They are growing rapidly, and I'm helping out with some high-level signal processing, project management, and company process to help them grow. I'm working there a few days a week, plus keeping busy as a Dad, and I also enjoy sailing my little 16-foot "trailer sailer" and riding my bike.

RS: A little more about the trailer sailer? Picture?

DR: Sure—it's a Hartley TS16 16-foot sailing boat popular in Australia and New Zealand. It lives at my home, and once a week I tow it down to the sea and have a day out with friends and family.

References

[1] J. Van Meggelen, L. Madsen, and J. Smith, *Asterisk: The Future of Telephony*, 2nd edition (O'Reilly, 2007), p. 201: https://ftp.openbsd.org/pub/OpenBSD/distfiles/9780596510480.pdf.

[2] Quick price comparison: https://www.voipsupply.com/.

[3] Oslec Echo Canceller: http://www.rowetel.com/?page_id=454.

[4] https://xorcom.com/product/voip-hardware-echo-cancellation/.

[5] http://www.rowetel.com/?page_id=454.

[6] Open Source Echo Canceller: http://www.rowetel.com/?p=18.

[7] http://www.rowetel.com/?p=128.

[8] See /usr/share/doc/dahdi/README.Debian.

[9] https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux-stable.git/log/drivers/misc/echo?h=linux-3.15.y.