Extending D-STAR with Codec 2

Antony Chazapis, SV9OAN
Heraklion, Crete, Greece
chazapis@gmail.com

Abstract – D-STAR was the first digital voice system designed by radio amateurs specifically for amateur radio use. However, it relies on a proprietary, patented codec, which has been the subject of controversy among amateur radio operators and regulatory authorities. In this paper, we present a protocol extension, that enables the optional use of Codec 2 in voice streams – an open-source and patent-free alternative to the default codec. The "D-STAR vocoder extension" has been implemented in a series of accompanying, open-source software projects, which focus on seamless interoperability with current D-STAR deployments. Transcoding voice streams between codec variants is possible with a custom version of the popular xlxd reflector. In addition, we introduce Estrella, a software-only, radio-like desktop and mobile application, for over-the-network communications.

1. Introduction

Digital Smart Technologies for Amateur Radio (D-STAR) is a digital voice and data standard designed for amateur radio use [1, 2, 3]. The research leading to the original specification [4], published in 2001, was funded by the Japanese government and led by JARL, in a joint effort with Japanese radio equipment manufacturers. D-STAR digital voice transmissions require less bandwidth than analog FM and provision a small percentage of the stream to be used for arbitrary data, like text messages or GPS information. D-STAR repeaters may have modules in several bands (most commonly VHF and UHF) and are capable of forwarding streams from one band to another, or through a special "gateway" module which "links" the repeater to some other Internet-based endpoint. Digital voice, once picked up by a repeater or a "hotspot" (a local, low-power, radio-to-Internet gateway), can be easily relayed over the network to other physical or virtual repeaters (called "reflectors"), allowing users to participate in pre-established or ad hoc worldwide talk groups [5, 6].

However exciting new technologies may be – especially to amateur radio operators, D-STAR was received early on with mixed reviews. Two were the most prominent causes of criticism: the availability of products from a limited set of manufacturers at a relatively high price point, and the use of a closed-source, proprietary codec for compressing raw, digitized voice into a low-bitrate stream and vice versa. Both points are still valid – almost 20 years after the introduction of the standard.

Most of the D-STAR equipment available is manufactured by ICOM, Inc., which almost solely supports and promotes D-STAR technology [7, 8] (actually, "D-STAR" is an ICOM registered trademark). And

while the protocol defines the packaging and modulation details, it does not define the format of additional data inside voice streams, routing protocols, and other relevant extensions that are necessary to support end-to-end D-STAR network deployments and associated applications. ICOM uses proprietary implementations of several D-STAR functions [9, 10], which throughout the years have been reverse-engineered, in order to enable building custom D-STAR radios, repeaters [11], and even faster and more efficient routing mechanisms [12, 13]. The current ability of D-STAR to support such a variety of configurations and transparently bridge to other digital voice systems (like DMR and System Fusion), has been largely driven by a large collection of open-source software and hardware, supported by a vibrant community of amateur radio operators, repeater builders/maintainers, and hardware homebrewers, that continuously devise and implement innovative solutions (a wealth of historical information is included in [14], while early practical examples are shown in [15, 16])

The result of this effort is that a fully operational D-STAR network backbone (repeaters, reflectors, as well as routing and bridging functions) can now be completely implemented using open software, running on low-cost, commodity devices. However, this is not the case with end-user radios. The proprietary codec used by D-STAR, which is available in the form of a hardware chip, renders do-it-yourself radios very difficult to construct. While there have been some software implementations that interface with a chip attached via USB on a remote machine over the network (like DVTool [17] or BlueDV [18]), users mostly prefer the convenience of a self-contained hardware radio – still with a limited choice of options at relatively high price points.

Advanced Multi-Band Excitation (AMBE), the codec used by D-STAR, was initially selected by the designers as the only practical option available at the time. There has been some discussion on what will happen when the respective patent held by Digital Voice Systems, Inc. will expire (if it is actually valid and has not already [19]), however an open implementation would also require a significant amount of information made publicly available by the company, which is most probably unexpected. In practice, variations of the AMBE codec are used in most well-known contemporary digital voice modes, so the codec selection is no longer a source of strong dispute; the proprietary codec issue has even resulted in a government ban of the mode all together in France. An alternative to AMBE, Codec 2, has been available for some time now. Codec 2 is open-source and patent-free, capable of compressing speech to very low bit-rates [20]. Moreover, it has already been used in amateur radio applications, most notably as an integral part of FreeDV, a digital voice mode for HF [21].

In this paper, we present a protocol extension to enable the optional use of Codec 2 instead of AMBE in D-STAR communications. The possibility of such an implementation had been discussed in the past [22], but – to our knowledge – had never been realized. The open source codec, will hopefully address both aforementioned issues, enabling the emergence of affordable, easy-to-build D-STAR radios, covering a wide spectrum ranging from software-only solutions connecting directly to reflectors, to completely independent hardware devices. Having D-STAR endpoints with no hardware requirements may also

allow an entirely new range of network-based applications, either stand-alone, or as interfaces to existing, external systems.

The changes to the protocol, along with their implications, are discussed in detail in the following chapter. In summary, a currently unused header flag is employed to mark the codec type in voice frames. This work is not meant to replace current D-STAR deployments, but augment them in a fully compatible way. To this end, we have implemented a reflector (based on xlxd [23]) that is capable of transcoding between AMBE and Codec 2 streams. The reflector is part of a series of extension-compatible, open-source software solutions, which also includes a set of command line utilities to experiment with the extension and a software-only client, called "Estrella", available in desktop and mobile versions.

2. Design

The D-STAR protocol specification defines the bit framing used to transmit either data or voice streams. Each stream begins with a synchronization pattern, followed by a header, and – in the case of voice communications – a variable number of alternating voice and data frames (Fig. 1) [24].

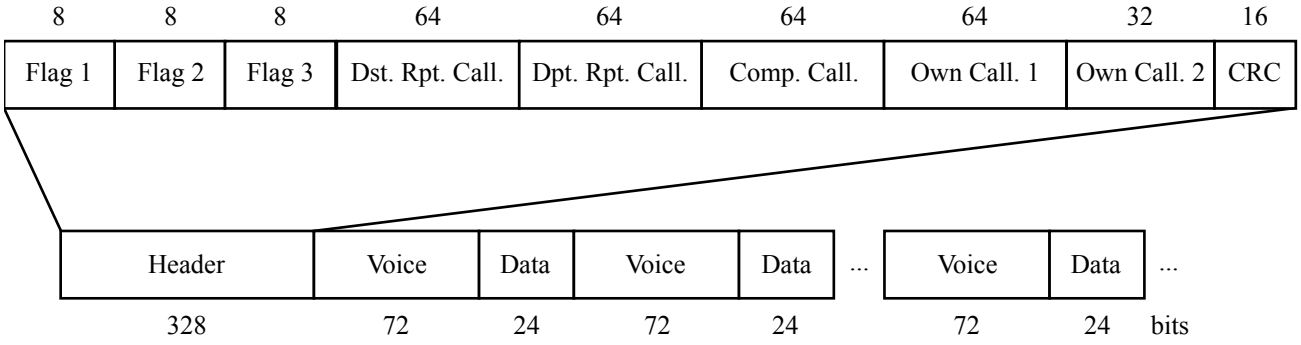


Figure 1: Overview of a D-STAR voice stream

The header, which is 328-bits long carries primarily addressing and routing information: who is sending the stream, where it is headed, through which repeater or gateway, etc. (It is actually coded into 660 bits when transmitted, after bit interleaving and scrambling.) Except callsigns and 2 trailing bytes for the checksum, the header also includes 3 distinct 1-byte long "flags" for identifying the type of communication, triggering control functions, and – what is of particular interest here – protocol expansion. Specifically, "Flag 3" has by default all 8 bits set to zero; any other value is undefined and left for future use.

The 72-bit long voice frames carry the data from the AMBE codec, while the 24-bit long data frames which are interleaved between them, help in synchronization and are commonly used to provide redundant copies of the header, text messages, and other information, like sender location. The AMBE codec encodes speech at 3600 bps, including error-correction information. Thus, each 72-bit voice frame (9 octets) corresponds to exactly 20 ms of voice. No other bits in the D-STAR stream are relevant to voice information.

In the following paragraphs, we will present in detail how the unused bits of Flag 3 can be exploited in order to extend the D-STAR protocol with Codec 2 support. We propose setting specific bits in Flag 3 to signify different content types of the voice frame. Two variants are supported: one with higher voice quality but no forward error correction (FEC), and another one with slightly higher voice compression, but with additional error-correction data piggybacked at the end of each encoded voice chunk.

2.1. D-STAR vocoder extension

The "D-STAR vocoder extension" uses the Flag 3 byte of the header, to mark the vocoder type in the voice frames as follows (in accordance to section 2.1.1, page 4, of the D-STAR specification):

Bit	Meaning	Function
`0000000x`	Vocoder	`0`: AMBE (backwards compatible) `1`: Codec 2
`000000x1`	Mode	`0`: Codec 2 3200 (160 samples/20 ms into 64 bits) `1`: Codec 2 2400 (160 samples/20 ms into 48 bits) plus FEC (22 bits)
`00000100` to `11111111`	Undefined	Use for future expansion

Table 1: Possible values of Flag 3 with the D-STAR vocoder extension

Toggling the least-significant bit of Flag 3 switches between backwards-compatible AMBE mode and the new Codec 2-based voice frames. A second bit controls the bitrate of the codec and the presence of FEC.

Codec 2 has several modes, the highest bitrate one working at 3200 bps – a close match to 3600 bps. Note, however, that the Codec 2 algorithm only provides speech processing and the resulting stream does not include any error-correction. 20 ms of voice in Codec 2 3200 mode require 64 bits (8 octets), which fit in the 72-bit space available, but do not leave much space for implementing FEC. The next available Codec 2 mode, which operates at 2400 bps, requires just 48 bits (6 octets) to encode a speech waveform of equal duration, which leaves an adequate amount of bits free, to protect the first 24 bits of the voice data with two applications of the (23, 12) Golay code. Each application produces an additional 11 bits that are used for error correction at the receiving side, raising the grand total of voice-related bits to 70; 2 short of the space available. (In practice, Codec 2 2400 mode also has 2 spare bits, so the transmission really requires 68 bits.)

This technique of employing FEC, is similar to what is implemented by FreeDV, a digital voice mode built upon Codec 2: FreeDV 1600 mode uses Codec 2 1300 mode to encode 40 ms of speech to 52 bits of voice data and then applies a (23, 12) Golay code to protect a 12-bit selection of those 52 bits. In Codec 2-encoded voice samples, some bits are more "important" than others. In the context of the implementation discussed here, it has been decided to apply the Golay code to the the first 24 bits of the

resulting data chunk encoded with Codec 2 2400 mode, which contains all the voicing and pitch/energy bits, plus the first 14 bits of harmonic magnitudes (Line Spectrum Pairs – LSPs).

2.2. Discussion

The proposed protocol extension is backwards compatible, so traffic using Codec 2 will pass through current D-STAR hardware (repeaters, hotspots, etc.) and software (repeater controllers, reflectors, etc.). Of course, hardware transceivers with AMBE chips are incompatible and cannot be used to directly communicate with Codec 2 extension implementations.

Interoperability between vocoder modes can be established using Internet-based D-STAR reflectors, in a similar fashion that they are currently being widely used to transcode and bridge between D-STAR, DMR, and System Fusion. In particular, in the next chapter we will discuss how xlxd, which is the most widespread software used in this setup, has been extended to allow transparent communications between AMBE-based devices and Codec 2-based, software-only clients. Note that although early tests have confirmed that xlxd is indeed compatible to pass through Codec 2-based transmissions without any changes, we have selected to apply a different connectivity setup for clients that support the extension (on a different UDP port using the familiar DExtra protocol [25], which has been named "DExtra Open"), in order to avoid user confusion and establish a fully compatible path for trouble-free adaptation of the new mode. Codec 2-based clients are thus "isolated" from their AMBE-only counterparts and respective streams are properly transcoded in order to allow cross-codec interaction.

The wide-spread use of Internet-based digital voice, primarily in the form of talking through hotspots, but also via desktop/mobile applications – like DVTool/BlueDV, is the main reason it has been decided to allow selecting between the 2 variants of Codec 2. For over-the-Internet communications, FEC is not really necessary, as malformed UDP packets – however rare – are dropped by the networking layer and never reach the application. It is therefore advised to always use Codec 2 3200 mode in such setups and enjoy the slight increase in voice quality.

In fact, we expect that dropping the AMBE codec from the protocol, hence the requirement of interfacing with a hardware chip, will allow a new generation of software-only clients to be implemented. Such desktop or mobile clients could directly communicate with each other, but to keep up with the usage paradigm of current radio-based transmissions, they will probably connect to some centralized "hub" (a reflector), that will allow broadcasting voice streams to all directly or indirectly linked users. In the next chapter, we also present Estrella, a D-STAR software implementation that allows communicating through existing reflectors without the need of any AMBE hardware.

The open source codec, may also allow home-brewing transceivers using a Rasbperry Pi, and an attached radio, either in the form of a directly connected HAT (Hardware Attached on Top) [26], or an MMDVM modem [27] (even one constructed with through-hole components [28]) interfaced to a "traditional", analog radio. The necessary software could run on the Raspberry Pi itself, assuming a method to

attach a microphone and speaker, or – more likely – on a mobile device running an Estrella variation that instead of transmitting streams directly over the Internet, sends them to the Raspberry Pi over Bluetooth (and vice versa), using an established D-STAR over-the-network protocol like DExtra. Such a setup, which has the user interface separate from the radio instance, may allow a variety of interesting, interoperable implementations, that may in turn cover a multitude of different usability and physical deployment requirements.

3. Implementation

The D-STAR vocoder extension has been implemented in three separate, but interoperable, projects:

- pydv: A Python library and an associated set of executable command-line utilities to interface with D-STAR reflectors and transcode saved voice streams, either locally, or using the transcoding server employed by xlxd (ambed).
- chazapis/xlxd: A "fork" of the leading software used to implement D-STAR reflectors, enhanced with the ability to recognize and transcode Codec 2-based streams to AMBE and vice versa when the appropriate hardware is present. Used as a communications hub for D-STAR software clients implementing the extension and a bridge between such clients and devices (repeaters or hotspots) serving AMBE-only transceivers.
- Estrella: A software-only D-STAR application that implement Codec 2-based communications via a compatible reflector (like the aforementioned xlxd fork). Estrella is available for macOS and iOS, while an Android version is in the works.

All the projects above are open source (licensed under GPL) and freely available at GitHub [29, 30, 31, 32]. Further details for each are presented in the following paragraphs.

3.1. pydv

The pydv project was originally developed as a loose collection of code to assist in the early stages of development of the extension. Now – at version 2.0 – it provides Python interfaces to manage DExtra and DPlus connections, convert from network data to D-STAR streams and vice versa, save and load such streams to/from .dvtool files [33], as well as encode and decode voice data using Codec 2 or AMBE (decode only via mbelib [34]), and transcode via a compatible ambed server (included in chazapis/xlxd).

The following command-line executables are included:

- dv-recorder, which connects to a reflector and records traffic in .dvtool files
- dv-player, which plays back a .dvtool file to a reflector
- dv-encoder, which converts a .wav file to a .dvtool file using the Codec 2 vocoder
- dv-decoder, which converts a .dvtool file using any vocoder to .wav
- dv-transcoder, which connects to an ambed server and converts a .dvtool file using the AMBE vocoder to a .dvtool file using the Codec 2 vocoder and vice versa

These utilities can prove useful to anyone experimenting with D-STAR in general. As a library, pydv may support higher-level applications that require D-STAR reflector connectivity using any of the supported protocols. pydv is written in Python 2.7. Future work includes porting it to Python 3, as well as implementing more digital voice network protocols.

3.2. chazapis/xlxd

xlxd is the most modern D-STAR reflector software. In essence, the function of a reflector is simple: it provides network protocol handlers for clients to connect and submit digital voice streams to specific "modules" (connection contexts, virtual "rooms"). The reflector relays each stream, in real time, to any client connected via any protocol to the same module. Some reflectors can even connect as clients to other reflectors, organizing the network traffic in cross-country or cross-continent meshes, and enabling ad hoc linking of talk groups together based on group-specific policy and habits [35].

There are various already established D-STAR network protocol variants. xlxd implements them all, thus allowing interoperability between them. As such, it can be deployed as either a DPlus, DExtra, or DCS-compatible reflector, using REF, XRF, or DCS callsign prefixes respectively (or all of the above simultaneously). By advertising a reflector installation to the appropriate callsign-to-IP resolution registries, clients can use standard D-STAR commands to establish repeater/hotspot links to the reflector. Moreover, xlxd introduces the notion of cross-reflector "trunks" (using the XLX protocol), which convey the streams of many modules simultaneously, in practice "extending" part of one reflector from one installation to another. A group of linked reflectors is commonly called a "constellation".

Recently, xlxd has also been interfaced to similar DMR network structures, like the BrandMeister network [36]. DMR uses a variant of AMBE to encode/decode speech in transceivers, so the streams running through xlxd in that case are not directly compatible with D-STAR. To cross codec barriers, a secondary software service called ambed, provides transcoding between the two formats. ambed requires the presence of one or more hardware chips to decode data from one AMBE variant into voice and encode it into the other. AMBE chips may have one or more channels; at least two are required to transcode a stream in real time.

Given the success and momentum of xlxd, as well of the eagerness of its developers to support any new and exciting technologies in the digital voice arena, it seemed as the perfect candidate to base the implementation of the D-STAR vocoder extension interoperability with other modes. As discussed, because of the extension's backwards compatibility, xlxd could be deployed as-is to support Codec 2-based clients using any existing D-STAR network protocol, but such an arrangement would magnify incompatibility issues. Instead it was decided to work first – before even starting implementing clients – in the direction of providing a unified substrate for codec-independent communications.

To this end, the xlxd/ambed project was forked and patched as necessary to transcode and bridge voice streams of any existing format to the new extension. The solution implements an additional DExtra lis-

tener on a different port (30201 instead of 30001). The new port is to be used by reflectors with the "ORF" callsign prefix (Open ReFlector). Any client connected to an ORF reflector will receive streams encoded with Codec 2. All other protocol handlers will still send out data encoded with AMBE. Note that the protocol/port only affects data transmitted by the reflector. The stream codec is recognized by all protocol handlers, so a client can still transmit data using any codec on any port. The rationale behind this is that DExtra links may be used by repeaters or other reflectors, so it is not really possible to know what their clients support. So, nothing will change when linking a repeater to an XRF reflector, but will do when linking to an ORF one. The new port endpoint has been named "DExtra Open" to distinguish it from the protocol running on the default DExtra port.

A detailed outline of the code changes done is attached to the pull request that has been submitted upstream, so the work will eventually be included into future versions of xlxd. In summary, most of the effort was concentrated into ambed, so each "vocodec" channel will function with three interfaces (1 in/2 out, instead of 1 in/1 out). Each such channel now has a (virtual) Codec 2 interface attached, along with the two physical ones for AMBE. When a new incoming stream presents itself, the appropriate transcoding channel is selected depending on the input codec. Channels are grouped together in triplets. Each group contains all possible permutations of respective interfaces – three channels, while only one channel from each group can be used at a given time. Groups represent hardware resources that cannot be shared between streams; channels represent input/output configurations. The rest of the work involved interfacing xlxd to the new "1 codec in, 2 codecs out" ambed interface, the new DExtra Open protocol listener on port 30201, etc. xlxd and ambed are written in C++.

The latest version of chazapis/xlxd is currently running at XLX393/XRF393 [37], which is deployed on a Raspberry Pi and supports transcoding a single stream at a time by employing two DVMEGA DV-stick30 USB interfaces (Figure 2). Operation is not restricted in any way, and it can be freely used for testing and experimenting with new digital voice technologies. XLX393/XRF393 is also reachable via DMR, through TG20209 of the BrandMeister network.

Future work includes deploying an ORF registry to track corresponding reflector deployments. The ORF registry will be useful for Codec 2-based software clients, in order for them to display compatible and available servers for connecting to. Assuming Codec 2-based D-STAR hardware devices at some point, the ORF registry may also tie in to repeater/hotspot software (like the ircDDB Gateway), to enable routing respective linking requests to the appropriate DExtra Open network endpoints.

Figure 2: XLX393/XRF393 current hardware setup

3.3. Estrella

Estrella is a radio-like software-only client for DExtra Open reflectors. Two platform variations have been implemented: a desktop-compatible macOS version and a mobile-friendly iOS one. Both use a similar user interface and aim for a very simple user experience: the user enters the connectivity details (callsign and ORF network endpoint), and is then presented with a minimalistic screen showing the connection status and providing a PTT button to initiate transmissions (Figures 3, 4). Although direct client-to-client connections could be possible, it has been decided to only support reflector-based setups, in order to match as close as possible the practice of using a "traditional" RF radio transceiver, encourage the deployment of new reflector software that complies with the proposed vocoder extension, and foster an environment of interoperable cross-codec digital voice applications and devices. As mentioned, with chazapis/xlxd installed and the appropriate hardware, the reflector will handle transcoding and bridging streams of different types and protocols.

Estrella implements the D-STAR vocoder extension and uses a linked Codec 2 library to encode and decode D-STAR communications without the need of any external hardware. macOS and iOS versions are both are written in Objective-C, and share a significant amount of code that is abstracted into 2 libraries (called Frameworks in Objective-C nomenclature) [38, 39]:

- CocoaDV, a Cocoa Framework to manage DExtra connections to D-STAR ORF reflectors
- CocoaCodec2, a Cocoa Framework for the Codec 2 low bit-rate speech codec

CocoaCodec2 bundles the source code of the Codec 2 SVN repository in an Xcode-friendly format. (details on how this was accomplished are given at its GitHub page). CocoaCodec2 is used by CocoaDV,

which in turn provides the necessary network and digital voice stream abstractions to Estrella. The result is that the core Estrella implementation only needs to concentrate on graphical user interface elements, handling connectivity events, stream buffering, and managing the available audio hardware. Actually, as the macOS and iOS core libraries have similar APIs, the respective Estrella codebases are also very similar.

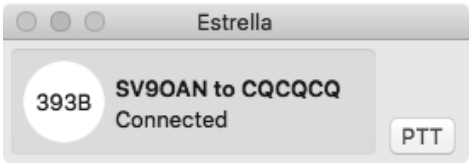


Figure 3: Estrella's main window (macOS version)

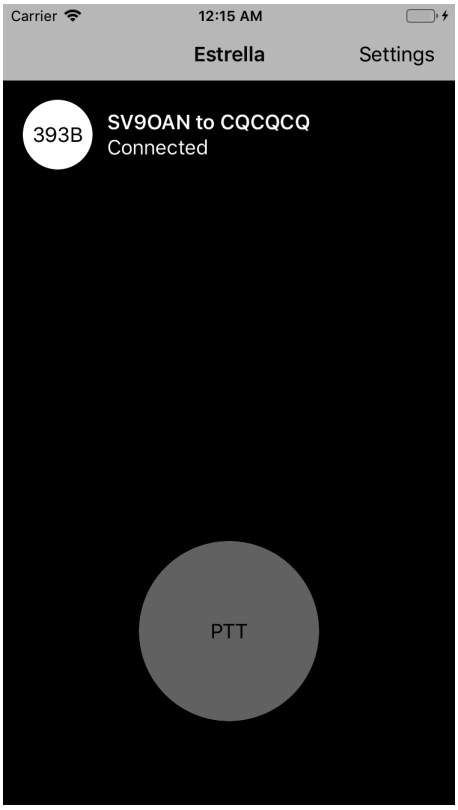


Figure 4: Estrella's main window (iOS version)

Current plans include completing the Android variation of Estrella and releasing all binary forms on corresponding online distribution services (a binary macOS version is already available for download at GitHub). Future versions may present a simpler user interface for configuration, by integrating server selection with the ORF registry, and incorporate text messaging and D-PRS functionality (exchanging positional data and presenting it on a map, as done by APRS [40]).

4. Conclusion

In this paper, we have described the design and implementation of an extension to the D-STAR protocol that uses Codec 2 for voice communications, instead of – or alongside – AMBE. We hope this work to become a significant milestone in D-STAR evolution, as it enables the development of completely open end-user radios and services, seamlessly interoperating with any current deployments. Moreover, the method used here can support other, similar D-STAR extensions, in order to introduce new voice codecs, or other, more radical changes to the stream structure.

In practice, in nearly 20 years of existence, D-STAR has already evolved in many ways. However, the available protocol specification does not include many details that are now considered *de facto*. Back then, it was decided to leave out some implementation specifics: the structure of text messages or location information included in voice streams, the routing techniques and associated callsign discovery methods, the format of streams when relayed over the network, etc. Many of these functions have been implemented in proprietary extensions, that – although mostly reverse-engineered – are for obvious reasons poorly documented. Discovering how D-STAR radios, repeaters, gateways, and reflectors work is a formidable task, as anyone interested can only find information in scarce Internet resources – that may vanish at any point – or deep into the codebase of related software projects. This unfortunate fact, neither matches the open nature of amateur radio, nor aligns to the (presumably) original intention of the D-STAR design group: to publish an open document that will provide the basis of cooperation and interoperability between radio amateurs and equipment manufacturing companies.

Adoption of digital radio technologies is growing larger every year, so D-STAR is now more relevant than ever. Both as a standard for digital radio transmissions, and as the foundation of an Internet-based, worldwide communication network for amateur radio. Sooner rather than later, JARL – or any other big and respectable amateur radio association – should convene a new standards body to release a revised, modern D-STAR standard, documenting established usage of the protocol and providing new directions for the future. If that ever happens, hopefully an open voice codec will indeed be an option – either the one presented here, or any other patent-free solution; in the true spirit of amateur radio.

Acknowledgements *– This work would not even be possible without the tremendous effort David Rowe, VK5DGR, has put into creating and maintaining Codec 2. David was also kind enough to discuss aspects of this project with me when I contacted him, asking for suggestions on how to move forward with FEC implementation. D-STAR would never reach the momentum it has today without the software written by Jonathan Naylor, G4KLX, Jean-Luc Deltombe, LX3JL, and many others. Their code served as an invaluable resource of knowledge – an insight into D-STAR and digital voice in general. I express my personal gratitude to all, as well as Bruce Perens, K6BP, for always trying to steer the world in the right direction.*

Notes *– D-STAR is a registered trademark of ICOM, Inc. AMBE is a registered trademark of Digital Voice Systems, Inc.*

References

[1] John Gibbs, KC7YXD, "D-STAR: New Modes for VHF/UHF Amateur Radio, Part 1", *QEX*, Jul/Aug 2003, p. 30–34. http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=604

[2] John Gibbs, KC7YXD, "D-STAR, Part 2: Design Considerations", *QEX*, Sept/Oct 2003, p. 22–28. http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=605

[3] John Gibbs, KC7YXD, "D-STAR, Part 3: Implementation", *QEX*, Noc/Dec 2003, p. 42–47. http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=606

[4] "D-STAR System". https://www.jarl.com/d-star/shogen.pdf

[5] Toshen M. Golias, KEØFHS, "Amateur Radio Notes: Diving into D-STAR", https://amateurradionotes.com/d-star.htm

[6] Toshen M. Golias, KEØFHS, "Amateur Radio Notes: Hanging out with hotspots", https://amateurradionotes.com/hotspots.htm

[7] ICOM, Inc., "D-STAR System Introduction". http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=447

[8] ICOM, Inc., "D-STAR. For the Second Century of Amateur Radio". http://www.icomamerica.com/en/downloads/DownloadDocument.aspx?Document=366

[9] Peter Loveall, AE5P, "D-STAR Uncovered", *Proceedings of the 27th ARRL and TAPR Digital Communications Conference*, Sept 2008. https://www.tapr.org/pdf/DCC2008-D-STAR-AE5PL.pdf

[10] Jonathan Naylor, G4KLX, "The Format of D-Star Slow Data". https://www.qsl.net/kb9mwr/projects/dv/dstar/Slow%20Data.pdf

[11] Jonathan Naylor, G4KLX, "g4klx/DStarRepeater: The D-Star Repeater.", https://github.com/g4klx/DStarRepeater

[12] "ircDDB". http://db0fhn.efi.fh-nuernberg.de/doku.php?id=projects:dstar:ircddb

[13] Jonathan Naylor, G4KLX, "g4klx/ircDDBGateway: The ircDDB Gateway for D-Star", https://github.com/g4klx/ircDDBGateway

[14] Steve Lampereur, KB9MWR, "Digital Voice applications and Ham Radio". https://www.qsl.net/kb9mwr/projects/dv/plan.html

[15] John Hays, K7VE, "D-STAR for the Technically Curious", *Presentation at the 29th ARRL and TAPR Digital Communications Conference*, Sept 2010. https://www.tapr.org/pdf/DCC2010-D-STAR-technical-K7VE.pdf

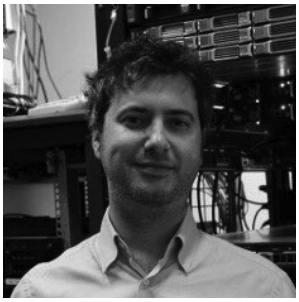
[16] Mark Braunstein, WA4KFZ, "Introduction to D-STAR Basics", *Presentation at the 30th ARRL and TAPR Digital Communications Conference*, Sept 2011. http://www.tapr.net/pdf/DCC2011-Intro_to_D-Star-Mark_Braunstein_WA4KFZ.pdf

[17] "DV Dongle". http://www.dvdongle.com

[18] "BlueDV". https://www.pa7lim.nl/bluedv/
[19] Bruce Perens, K6BP, "AMBE Exposed", *Presentation at the 33rd ARRL and TAPR Digital Communications Conference*, Sept 2014. https://www.qsl.net/kb9mwr/projects/dv/codec/AMBE_Exposed.pdf
[20] David Rowe, VK5DGR, "Codec 2 - Open Source Speech Coding at 2400 bit/s and Below", *Proceedings of the 30th ARRL and TAPR Digital Communications Conference*, Sept 2011. http://www.tapr.net/pdf/DCC2011-Codec2-VK5DGR.pdf
[21] "FreeDV: Open Source Amateur Radio Voice". https://freedv.org
[22] Bruce Perens, K6BP, and David Rowe, VK5DGR, "Codec2: An Open Future for Digital Voice", *Presentation at the 29th ARRL and TAPR Digital Communications Conference*, Sept 2010. https://www.tapr.org/pdf/DCC2010-Codec2-presentation-K6BP-VK5DGR.odp
[23] Jean-Luc Deltombe, LX3JL, "LX3JL/xlxd: HAM radio multiprotocol dstar reflector server". https://github.com/LX3JL/xlxd
[24] Denis Bederov, DL3OCK, "DSTAR radio frame structure in DV mode". http://db0fhn.efi.fh-nuernberg.de/lib/exe/fetch.php?media=projects:dstar:ircddb:dstar_dv_frame3_en.pdf
[25] Steve Lampereur, KB9MWR, "D-Star Open Source / Dextra project by Scott Lawson, KI4LKF". https://www.qsl.net/kb9mwr/projects/dv/ki4lkf/ki4lkf.html
[26] Mathis Schmieder, DB9MAT, "mathisschmieder/MMDVM_HS_Hat: MMDVM_HS Hat for the Raspberry Pi (Zero)". https://github.com/mathisschmieder/MMDVM_HS_Hat
[27] Jonathan Naylor, G4KLX, "g4klx/MMDVM: The firmware for the MMDVM (Multi-Mode Digital Voice Modem)". https://github.com/g4klx/MMDVM
[28] Florian Wolters, DF2ET, "Handcrafted MMDVM Adapter", https://www.florian-wolters.de/blog/2016/02/25/handcrafted-mmdvm-adapter/
[29] Antony Chazapis, SV9OAN, "chazapis/pydv: D-STAR library and utilities in Python". https://github.com/chazapis/pydv
[30] Antony Chazapis, SV9OAN, "chazapis/xlxd: HAM radio multiprotocol dstar reflector server". https://github.com/chazapis/xlxd
[31] Antony Chazapis, SV9OAN, "chazapis/Estrella-macOS: ORF reflector client for macOS". https://github.com/chazapis/Estrella-macOS
[32] Antony Chazapis, SV9OAN, "chazapis/Estrella-iOS: ORF reflector client for iOS". https://github.com/chazapis/Estrella-iOS
[33] Kristoff Bonne, ON1ARF, "Format of files and UDP-streams used on D-STAR". https://qsl.net/kb9mwr/projects/dv/dstar/formats%20of%20files%20and%20UDP-streams%20used%20on%20D-STAR.pdf
[34] "szechyjs/mbelib: P25 Phase 1 and ProVoice vocoder". https://github.com/szechyjs/mbelib
[35] Bob Scott, W6KD, "Reflections on Reflectors". https://qsl.net/kb9mwr/projects/dv/dstar/Reflections%20on%20Reflectors%201.02.pdf
[36] "BrandMeister". https://brandmeister.network

[37] "XLX393 Multiprotocol Gateway". http://reflector.rasc.gr
[38] Antony Chazapis, SV9OAN, "chazapis/CocoaDV: Cocoa Framework for D-STAR". https://github.-com/chazapis/CocoaDV
[39] Antony Chazapis, SV9OAN, "chazapis/CocoaCodec2: Cocoa Framework for Codec 2". https://github.com/chazapis/CocoaCodec2
[40] Peter Loveall, AE5PL, "APRS and D-STAR = D-PRS", *Proceedings of the 26th ARRL and TAPR Digital Communications Conference*, Sept 2007. https://www.tapr.org/pdf/DCC2007-D-PRS-AE5PL.pdf

About the author



Antony Chazapis, SV9OAN, is licensed since 2009. He is a member of RAAG, RASC, ARRL, and the volunteer group HARES (Hellenic Amateur Radio Emergency Service). Since 2017, he has been serving as the Association Manager for the SOTA award program in Greece. Antony enjoys DXing, contesting and exploring what makes digital modes tick. He is the author of PocketPacket, an APRS client application for macOS and iOS. His work on D-STAR includes Estrella, an open-source, software-only radio for macOS and iOS, and associated server software and utilities, that enable D-STAR communications using Codec 2. He holds a PhD in Computer Science and is currently employed by a Toronto-based tech startup implementing bleeding-edge storage solutions.