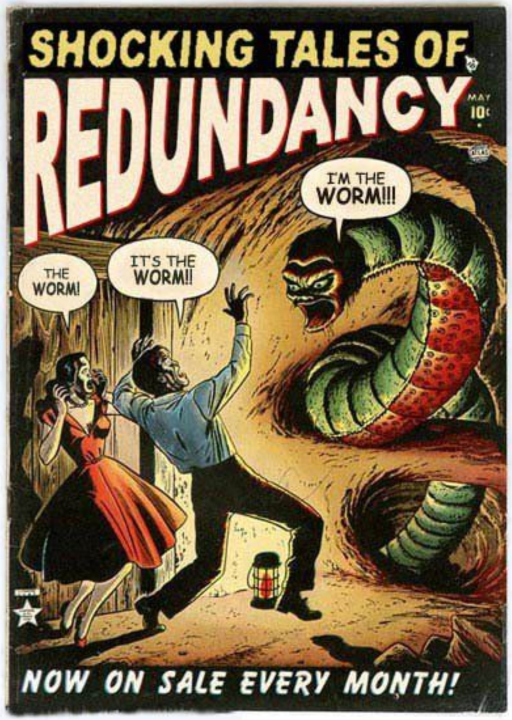
Protect your Bits: An introduction to gr-fec

FOSDEM '19, Free Software Radio Devroom

Martin Braun

Representin' Ettus Research & GNU Radio



Forward Error Correction 101

- In the 1940s, Shannon came up with most of the theory we use these days for wireless communications
- The Shannon-Hartley Theorem gives a hard upper bound on how much data can be transmitted over a point-to-point link (with AWGN interference)
- It doesn't say how, though!

THEOREM 2: *Let P be the average transmitter power, and suppose the noise is white thermal noise of power N in the band W. By sufficiently complicated encoding systems it is possible to transmit binary digits at a rate*

$$C = W \log_2 \frac{P + N}{N} \qquad (19)$$

with as small a frequency of errors as desired. It is not possible

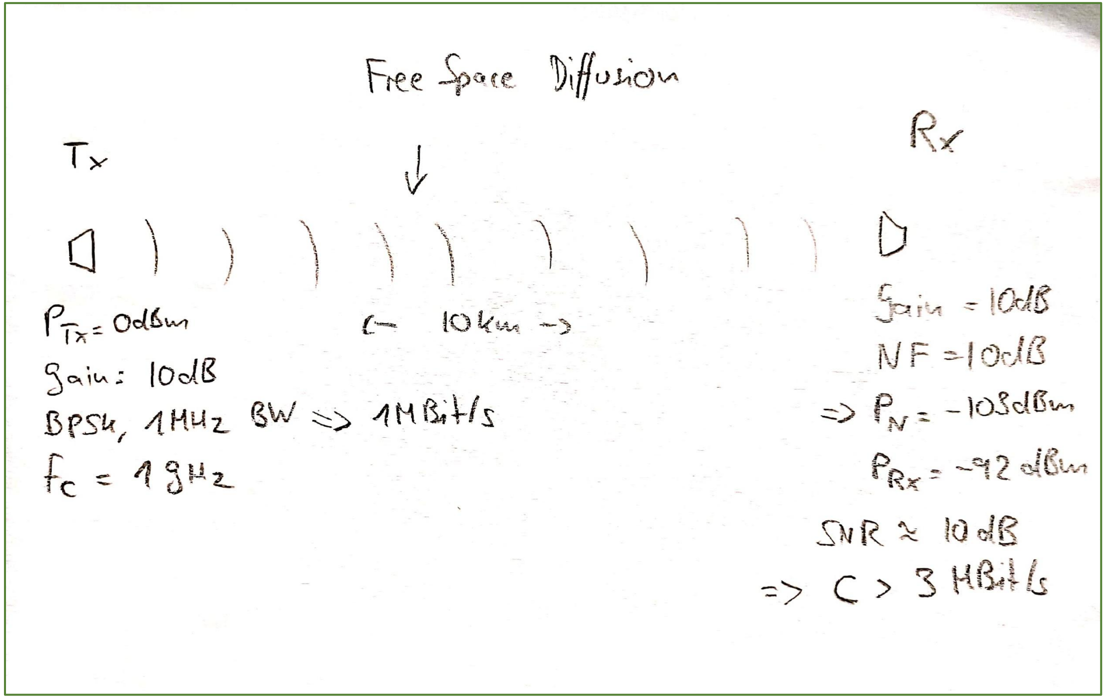
(1) C.E. Shannon, "Communication in the Presence of Noise", January 1949

GRC Examples for this Talk

- I’ll make sure they get uploaded to the FOSDEM website, if they’re not part of the tree
- I used maint-3.7 for this stuff. As of now, pre-3.8 has some issues with the GRC examples, (e.g., no bus ports) and I’m not going to risk this talk to test master branch...

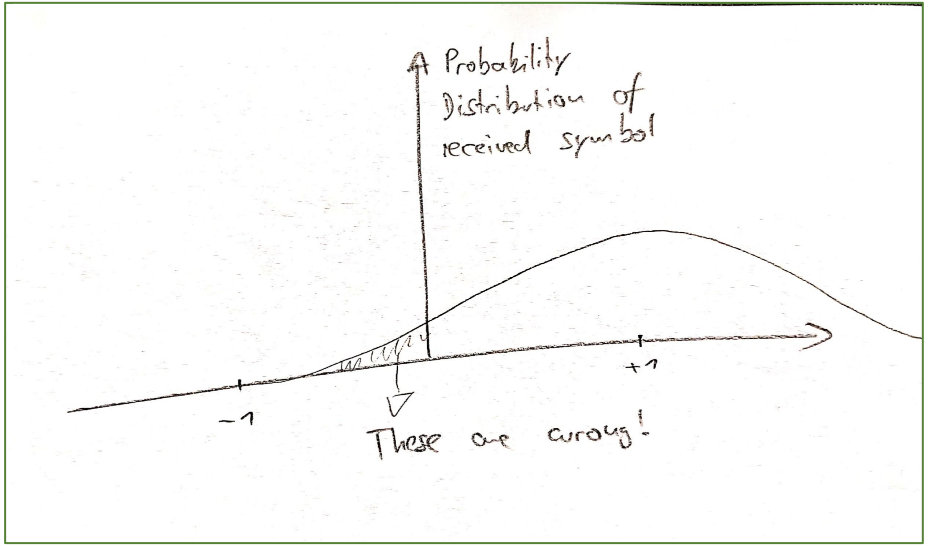
Let's try without coding!

- So can we just transmit below the Shannon limit?
  - uncoded.grc
  - We're more than a factor of 3 away from Shannon's limit



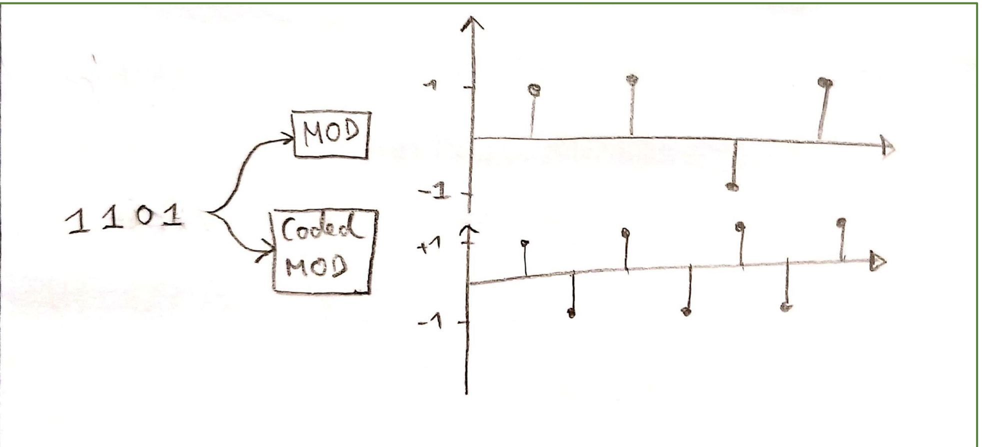
What went wrong?

- Let’s enable those disabled blocks
- All our decoder can do is get the sign of the bits, but noise will statistically ruin those
- Looks like our transceiver chain was not sufficiently complicated!



Let's add Redundancy

- As good ol' Claude says, we need to make our transceivers "sufficiently complicated"
- Core tenet of all FECs: Add more stuff in a structured fashion!
- Receivers can tell if a received sequence "makes sense"



- More bits! ("Code Rate equals 7/4")
- Different bits ("unsystematic")

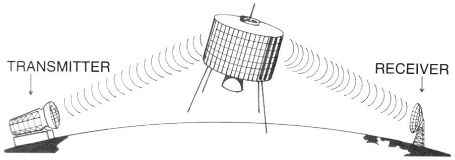
Concepts of FEC

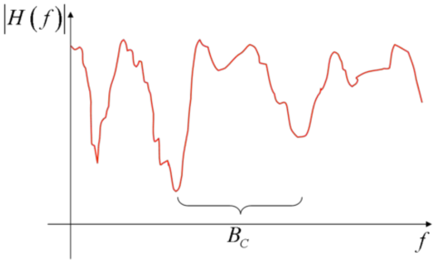
(This is where I fast-forward over several semesters worth of information)

- Systematic codes: Codes that contain the uncoded data
- Latency: Coding/Decoding can incur additional processing latency
- Interleaving/Concatenation: We might combine multiple codes in a smart way for additional benefits
- Coding gain: The actual advantage of using a code vs. transmitting uncoded data
- Puncturing: After adding redundancy, we can remove some of the bits again to scale the coding rate

Examples of FEC Applications

- Satcom
  - Low SNR, AWGN, small variances in SNR
- CD/DVD Drives
  - High SNR, bursty errors,
- Your LTE phone
  - Everything is bonkers, multi-path, Doppler, fading..





Noteworthy Codes

Ettus Research™
A National Instruments Brand

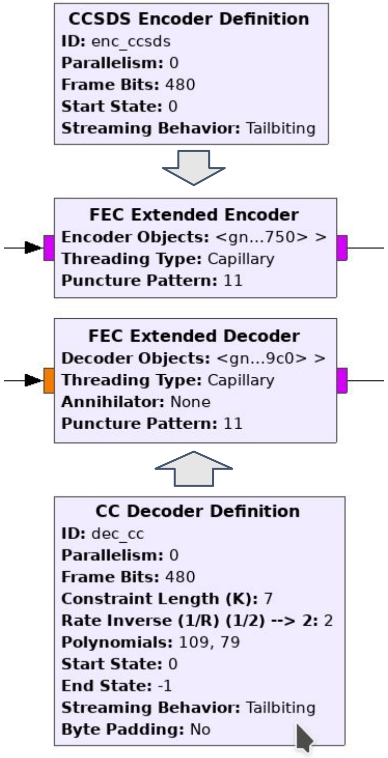
- Convolutional Codes (802.11a)
- Turbo Codes (LTE)
- Hamming Codes (Usually the first code you learn in school)
- POLAR Codes (5G NR)
- BCH Codes (CD/DVD Players)
- Reed-Muller, Reed-Solomon, ...
- There’s many.

Enter gr-fec

- gr-fec usually builds out-of-the-box with GNU Radio
  - Use –`DENABLE_GR_FEC=ON` to be certain
  - Requires VOLK
- gr-fec has a bunch of great examples, let’s check them out!
  - Let’s start with fecapi_decoders.grc

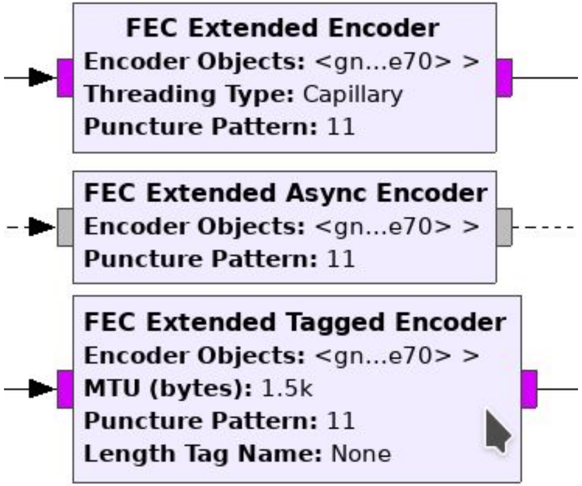
Blocks and Kernels

- Data streaming blocks are separate from the FEC implementations
- Blocks match the type of streaming model, the kernel matches the FEC that is requested

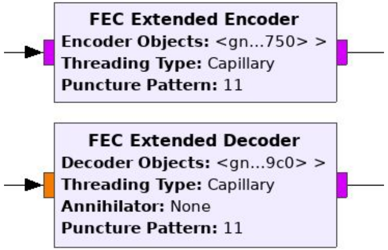


Block Types

- All blocks come in an “extended” variety: Added Python sugar for easier integration -> Use this in GRC unless you really know what you’re doing!
- Regular Encoder: Infinite-stream
- Async Encoder: For message passing applications
- Tagged Encoder: For (the beloved) Tagged Stream Blocks



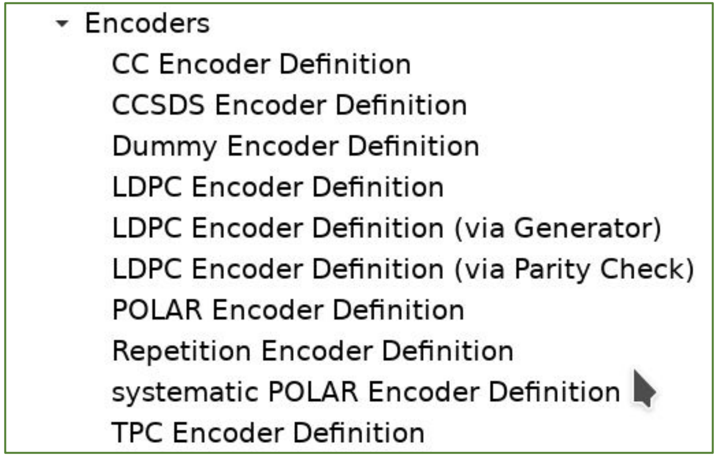
Block Settings & Functions



- Encoder Blocks consume and produce unpacked bits
- Decoder Blocks consume “soft bits” and produce unpacked bits
- Puncturing / Depuncturing is handled by the (extended) block (not the kernel)
- FEC blocks can be parallelized, the extended encoder will spawn multiple identical blocks in parallel

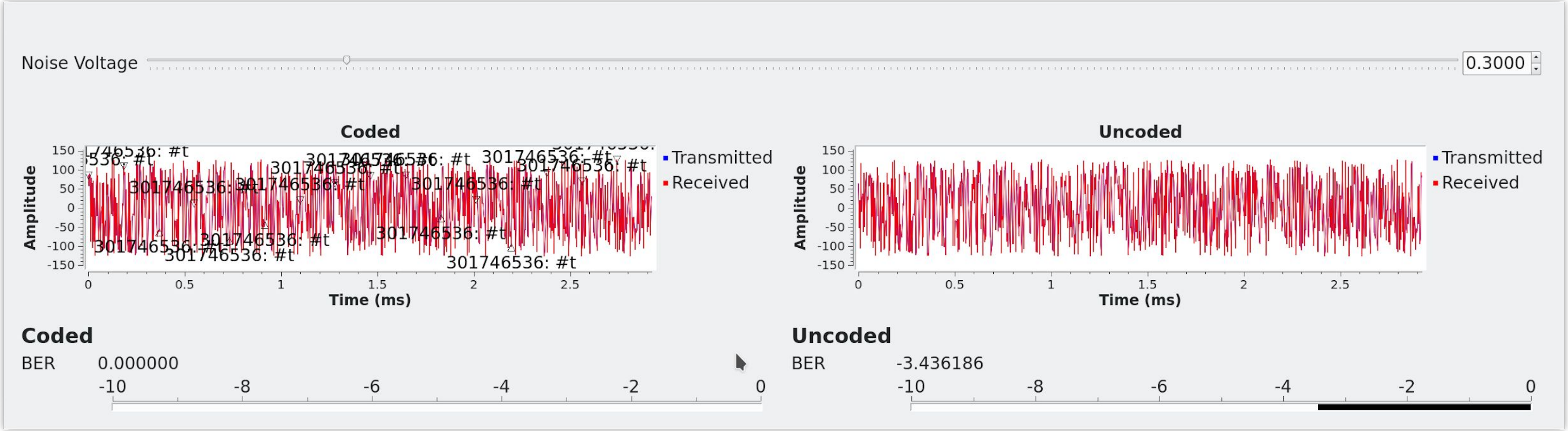
Available Kernels

- Dummy & Repetition Kernels for debugging and comparison
- Convolutional Codes
- LDPC Codes (various different implementations)
- Turbo Product Code
- POLAR Codes (various implementations)



Running gr-fec in the wild

- Let's check out polar_code_example.grc
- (Reminder: POLAR codes used in 5G NR control channels)



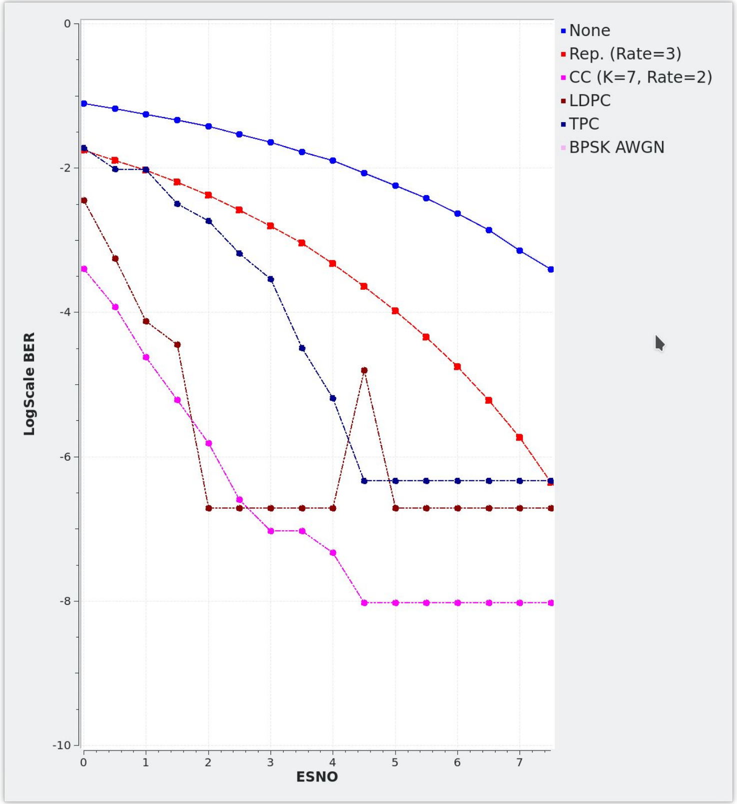
BER Simulations

Ettus Research™
A National Instruments Brand

- It’s debatable if GNU Radio is the right tool to do BER simulations, but you can test the capabilities of the kernels
- Unlike your typical scripted simulation, GNU Radio runs multiple AWGN channels at once
- Note: All of these examples require bus ports, wh
broken on 3.8/master as of 31-Jan-2019 (ple
- Note 2: There’s also a berawgn.py example, w
something else. Go check it out if you like.

BER Simulations

- It's still debatable if GNU Radio is the right tool to do BER simulations
- Make sure you interpret the results correctly! ES != EB. Low bit rates are hard to simulate.

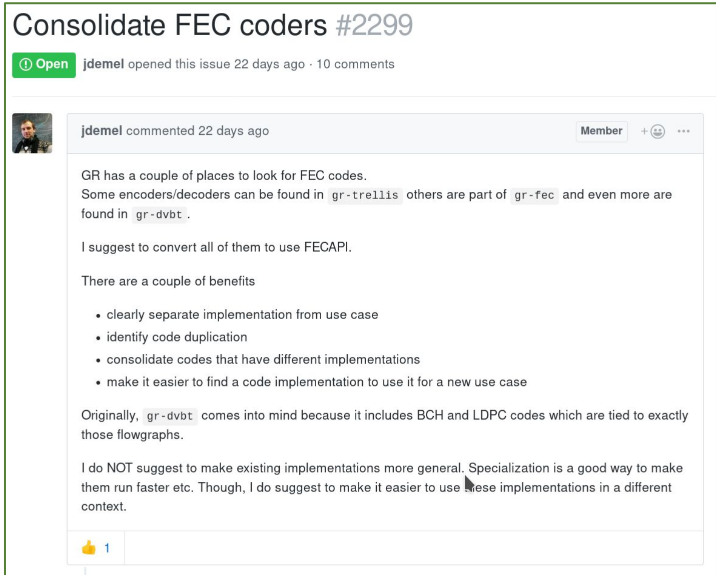


Pay our Respects!

- Thanks to Nick McCarthy for originally coming up with FECAPI (which became gr-fec)
- Johannes Demel, Manu TS, Tracy Perez, Tim O'Shea, Tom Rondeau: Noteworthy contributor of codes
- GRCon '16: SOCIS + POLAR Codes (J. Demel)
- GRCon' 16: POLAR Codes at hundreds of MBit/s (P. Giard)

Final Words

- FEC is the good kind of redundancy
- Let's stay modular -- let's re-use codes and set them free
- FEC is a critical and difficult part of wireless links. Having good, free implementations for those in GNU Radio is important for controlling our PHYs
- Join us in adding codes! We need to make them faster, and add more codes.



Consolidate FEC coders #2299

Open jdemel opened this issue 22 days ago · 10 comments

jdemel commented 22 days ago Member

GR has a couple of places to look for FEC codes.
Some encoders/decoders can be found in `gr-trellis` others are part of `gr-fec` and even more are found in `gr-dvbt`.

I suggest to convert all of them to use FECAPI.

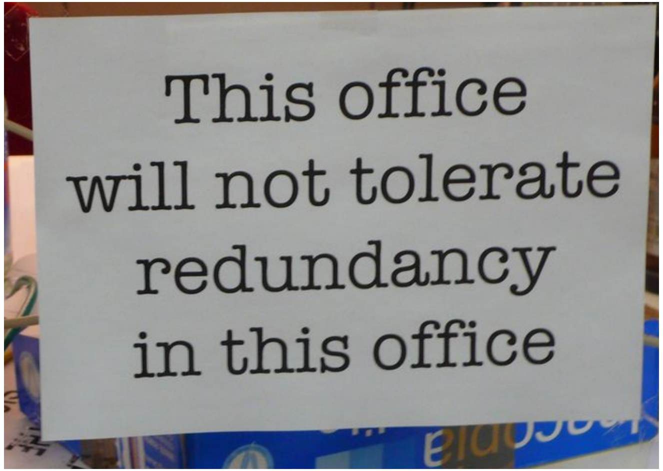
There are a couple of benefits

- clearly separate implementation from use case
- identify code duplication
- consolidate codes that have different implementations
- make it easier to find a code implementation to use it for a new use case

Originally, `gr-dvbt` comes into mind because it includes BCH and LDPC codes which are tied to exactly those flowgraphs.

I do NOT suggest to make existing implementations more general. Specialization is a good way to make them run faster etc. Though, I do suggest to make it easier to use these implementations in a different context.

👍 1


This office
will not tolerate
redundancy
in this office