Building a Moderately Complex Modem with Spare Parts

Dan CaJacob

Director of Space Systems
HawkEye 360, Inc.

GNURadio Conference 2017
Thursday 14 September 2017
Abstract

We present a QPSK modem with concatenated coding and IP encapsulation implemented entirely in GnuRadio. The modem is made with core GnuRadio components to the maximum extent possible, then relying on a few existing OOT modules and finally, after exhausting all other options, the author deigns to write a few custom blocks (in Python). Scary topics like constellation rotation/inversion correction and latency reduction are addressed in passing. Fun is had by all. An ill-advised live demo may be demanded. Hecklers point out that this is painfully close to a CCSDS recommended standard.
Overview

- Goal
- A Moderately Complex Modem?
- Spare Parts?
- Quick Success
- Modem
- A Few Custom Parts
- Five Stages of Avoiding Writing Code
- Defeating Latency
- Live Demo
- Future Work
- Acknowledgements
Goal

- Build a full-duplex modem with concatenated FEC codes corresponding closely to CCSDS recommended standard
- Support IP / UDP encapsulation
- Minimize latency
- Bonus: Add TCP acceleration via some PEP magic
A Moderately Complex Modem?

- Higher order modulation (QPSK)
- Concatenated Coding
  - ½ Rate Convolutional
  - RS + Scrambling and Interleaving
- Basically the CCSDS recommendation, without their insane mishmash of higher layers
- Phase Ambiguity Issues
- IP Encapsulation
  - Custom, toy protocol, but you could easily replace with HDLC, etc.
- Latency Reduction
- Do as much as possible in GRC
Spare Parts?

- Basically, I bend over backwards to not write a custom block (and fail)
- Because I’m lazy
- And because GNURadio largely makes this possible
- Also, there are some awesome OOT solutions out there
- And others that I completely missed
  - Sorry, gr-mapper
Quick Success

- Not too hard to build a modem with concatenated FEC codes
  - Thanks to awesome OOTs like André Løfaldli’s gr-ccsds
  - Experimented early with the old convolutional encoder block, quickly moved on to newer FEC API blocks
  - GR makes it easy to add Layers incrementally
- Use stock TUN block
  - Network interface to user-land code
- Latency was heinous (like 20 seconds)
  - Adding fill frames before modulator
- Works, but not really a good solution
Modem: Convolutional Code

- CCSDS recommends $R=\frac{1}{2}$, $K=7$
  - Each output bit is dependent on $2(K-1) = 12$ previous bits
  - So, we should ignore the first 12 bits sometimes
  - As a compromise for simplicity, ignore the first 16 bits
  - ASM is 32 bits, 64 bits after conv. code, so 52 of those are deterministically encoded
    - We just use 48 since it obeys byte boundaries
  - I’m actually not using the inversion yet, but Daniel Estevez made this easier to do, recently
Modem: Reed Solomon Code

- CCSDS recommends
  - RS (255, 223)
  - Interleaving depth = 5
  - Scrambling
- I use André Løfaldli’s gr-cssds OOT
  - Uses libfec (Phil Karn)
Modem: Transmitter

- **Virtual Source**
  - Stream ID: tun source

- **Constellation Modulator**
  - Constellation: <const_n QPSK>
  - Differential Encoding: No
  - Samples/Symbol: 4
  - Excess BW: 700m

- **CCSDS Encoder**
  - Length tag key: Yes
  - RS Encode: Yes
  - Interleave: Yes
  - Scramble: Yes
  - Idle Frames: False
  - Idle Block Time (ms): 50

- **Unpack K Bits**
  - K: 8

- **CCSDS Differential Encoder**

- **Pack K Bits**
  - K: 8

- **FEC Extended Encoder**
  - Encoder Objects: <qn..510> 
  - Threading Type: Capillary
  - Puncture Pattern: 11

- **UHD: USRP Sink**
  - Stream args: peshk=0.7
  - Device Address: type=b200
  - Samp Rate (Sps): 1M
  - Ch0: Center Freq (Hz): 400M
  - Ch0: Gain Value: 40
  - Ch0: Antenna: TX/RX
  - TSB tag name:
Modem: Receiver
Modem: IP Encap/Decapsulate Interface

TCP/UDP
- IP
  - LINK
  - IP ENCAP
  - PHY
  - CCSDS
  - RF

TCP/UDP
- IP
  - IP ENCAP
  - CCSDS
  - PHY
  - RF
A Few Custom Parts

- Tricky problems force me to write some code
  - Phase lock ambiguity
    - Spent a lot of time spinning my wheels on this
    - Eventually settled on a compromise solution that uses spare parts
  - Fill frame insert
    - Stupidly simple
    - Can probably do this with spare parts
  - Post Modulator, Convolutional Encoder Fill frame stitching
    - Not really custom code, but still hard to get right
  - IP Encapsulator / Decapsulator
    - Also stupidly simple
Five Stages of Grief: Avoiding Writing Code

1. Denial
   ○ Surely this exists...

2. Anger
   ○ Why the hell doesn’t this exist?

3. Bargaining
   ○ Can you help me make this exist?

4. Depression
   ○ I’ll never make this work.

5. Acceptance
   ○ OK, I’ll write some code.
   ○ OK, I’ll do the convolutional code by hand (Thanks for your help, Darek)

6. My own addition
   ○ Can somebody help me write this code?
Latency Reduction

● Add fill frames / symbols at the last possible moment
  ○ Really... for reals
  ○ It makes phase synchronization a lot harder (because of the convolutional code)

● But go ahead and try everything else first
  ○ GR buffer manipulation (nope)
  ○ USB buffer buggery (kinda nope)
  ○ Bargaining (nope nope)

● Smaller gains
  ○ Tweak convolutional encoder / viterbi decoder work length (meh)
  ○ Increase the data rate (at least until the Os come)

● Gr-perf-monitorx keeps you honest
Add Fill Frames Last

- Fill frames need to include all channel coding (convolutional coding!) and modulation
- If work is called and no items pending, send a fill frame
- Fill frame needs to be match up with previously sent data and data to come after (unpredictable)
  - Convolutional encoder and other blocks have memory, so this is hard!
  - But there's no better way to defeat latency

ASM Scrambled, Interleaved RS Data ASM Scrambled, Interleaved RS Fill Frame ... ASM ...

Convolutionally Encoded Data
Building the Fill Frame

- Fill frames are just a data block of 0s
- Several support flowgraphs
  - Idle_frame_encode
    - Encode fill sequence with RS, interleaving, scrambling, convolutional code
  - Sub graph in main flowgraph
    - Use Modulate Vector block to generate a modulated IQ vector
- Zero pad to flush, slice intelligently to combat memory in certain blocks
- \([0]^*223^*5 \rightarrow (-0.6960...0.9399...j), \ldots (-0.9150...+0.7737...j))\)
Building the Fill Frame

- Hand encode ASM
  - After much prodding and help from colleagues
  - Crucial to getting to bit-accurate generation
  - Generator blocks have some memory, so need to trim intelligently
Building the Correlation Sequence

- Several support flowgraphs
  - Asm_test
    - Encode ASM with convolutional code
  - Asm_post_costas
    - Map encoded ASM to complex symbols
- Zero pad to flush, slice intelligently to combat memory in certain blocks
- 0x1ACFFC1D → ((0.7071...-0.7071...j), … (0.7071...+0.7071...j))
Phase Synchronization

- Harder than BPSK
- Costas loop will lock, but perhaps not to the right rotation, order
  - Simulations can fool you (unless you simulate real world effects)
  - Over the air, you might be right 50% or 25% of the time, depending on coding
- Correlation estimator and 2nd Costas Loop clean up the ambiguity
- There are better ways to do this
Phase Synchronization

- There are better ways to do this

Table 2.4.11-1: Relationships Between the Transmitted and Received Data

<table>
<thead>
<tr>
<th>Carrier Phase Error (Degrees)</th>
<th>Received Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>I&lt;sub&gt;1&lt;/sub&gt;, Q&lt;sub&gt;1&lt;/sub&gt;</td>
</tr>
<tr>
<td>90</td>
<td>-Q&lt;sub&gt;1&lt;/sub&gt;, I&lt;sub&gt;1&lt;/sub&gt;</td>
</tr>
<tr>
<td>180</td>
<td>-I&lt;sub&gt;1&lt;/sub&gt;, -Q&lt;sub&gt;1&lt;/sub&gt;</td>
</tr>
<tr>
<td>270</td>
<td>Q&lt;sub&gt;1&lt;/sub&gt;, -I&lt;sub&gt;1&lt;/sub&gt;</td>
</tr>
</tbody>
</table>

Table 2.4.11-2: Summary of the Salient Features of the Preferred Techniques

<table>
<thead>
<tr>
<th>Available Techniques</th>
<th>Bit Error Rate (BER) Degradation</th>
<th>Advantages &amp; Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Word Detection</td>
<td>None</td>
<td>- Increase earth station complexity</td>
</tr>
<tr>
<td>Differential Data Formatting Without Forward-Error-Correction (FEC)</td>
<td>Increases by approximately a factor of two</td>
<td>- Simple to implement - Can cause degradation in the detection of the transmitted sync markers</td>
</tr>
<tr>
<td>Differential Data Formatting Inside the FEC Encoder and Decoder Pair (CODEC)</td>
<td>About 3 dB for convolutional code with R = 1/2, K = 7</td>
<td>- Provides quick phase ambiguity resolution - Requires overpowered link</td>
</tr>
<tr>
<td>Differential Data Formatting Outside the FEC Codec</td>
<td>Small</td>
<td>- Requires differential decoders at the station</td>
</tr>
</tbody>
</table>
IP Encapsulation / Decapsulation

- **Toy implementation**
  - Dead simple Python blocks
  - 1 IP / UDP packet per CCSDS frame
  - Just add a 2 byte header to define length of encapsulated packet
  - Enforce MTU size limit (2 header bytes + 1113 data bytes)

- **Better solutions**
  - HDLC encode to delimit packets

- **Worse solutions**
  - Actually follow CCSDS, don’t do that

<table>
<thead>
<tr>
<th>Header</th>
<th>IP Packet</th>
<th>Free (0 - n)</th>
</tr>
</thead>
</table>

ASM  | Scrambled, Interleaved RS Code blocks | ... |

...
Live Demo: Loopback Simulation

- Note unchanging fully encoded ASM symbols at start of frames
  - Accurate sticking of fully pre-encoded fill frames allows > 100x latency reduction
  - This was the principal challenge and success of this work
Live Demo: OTA

- Connect
- Ping
  - Highlight low latency
- Disconnect channel and demonstrate self-healing
- SSH / Mosh
  - Hollywood time!
- File Transfer?
- Demo PEP acceleration on a simulated high latency, high BER channel
Live Demo
Future Work

- Replace toy IP encapsulation protocol with HDLC / MPoFR
- Figure out how to synchronize OQPSK
  - Technically, CCSDS recommends OQPSK, not QPSK
- Replace hacky phase synchronizer with more robust and more efficient implementation
  - Maybe try gr-mapper?
- Fix bug with long-running file transfers
  - ASM seen in data packet, confusing PLL?
- Script the correlation and idle frame generation
- Measure BER vs. $E_s/N_0$
- Bonus: Get all the OOT stuff in tree
  - RS $\rightarrow$ FEC API, etc.
- Get an intern to do it all for me
Acknowledgements

- Darek Kawamoto
- Nick McCarthy
- Andy Walls & the IRC gang
- André Løfaldli
  - gr-ccsds
- Daniel Estevez
  - Read his blog: http://destevez.net/
- HawkEye 360
Questions?

- HE360 is hiring!

“We’d now like to open the floor to shorter speeches disguised as questions.”