Tools for High Sample Rate Recording and Post Processing: gr-analysis

Orin Lincoln and Paul Garver

GNU Radio Conference 2015
Team Background

- Intelligent Digital Communications is a Vertically Integrated Project team that involves a professor, graduate student, and a variety of undergraduates to explore a complex research problem.
- Our team uses software-defined radio techniques to explore RF spectrum utilization in extreme emitter-density environments, like the Georgia Tech football stadium.
Goal

- Collect large amounts of data from football stadium on a game day
- Analyze that data to make observations about how RF spectrum is used
  - Interference: cell phones transmitting WiFi and Bluetooth, coaches’ microphones, etc.
  - Localization of transmitters
- Draw conclusions, eventually
Our approach

- We started recording data files with a laptop, antenna, and USRP.
- Metadata included in filename
  - Ex: 0460_25_1315_SweepingFromNorthStandToSouthStand.sc16
Radio frequency sensor nodes (RFSN)

- Designed a solution with a network of spectrum sensing nodes in the football stadium
## Hardware setup

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ettus</td>
<td>782980-01 (B200)</td>
<td>RF Digitizer</td>
</tr>
<tr>
<td>Nat’l Instruments</td>
<td>783454-01</td>
<td>GPS Oscillator</td>
</tr>
<tr>
<td>Intel</td>
<td>BOXD54250WYK</td>
<td>Haswell i5 NUC PC</td>
</tr>
<tr>
<td>Samsung</td>
<td>MZ-MTE1T0BW</td>
<td>1TB Solid State Disk</td>
</tr>
<tr>
<td>Crucial</td>
<td>BLS2K8G3N169ES4</td>
<td>16GB DDR3 RAM</td>
</tr>
<tr>
<td>Nat’l Control Devices</td>
<td>R110PL ETHERNET</td>
<td>Ethernet Relay</td>
</tr>
<tr>
<td>L-COM</td>
<td>HG2458-20P</td>
<td>2.4/5GHz Antenna</td>
</tr>
</tbody>
</table>
Radio frequency sensing nodes (RFSN)

- Started deploying a network of nodes in the football stadium
 Subject to SWAP constraints
  - Size, Weight, Area, Power

 First one mounted on concession stand
Considerations

- We work with large data files from football games. The recordings may be several hours long.
- We also have multiple recordings of the same environment from different locations. To synchronize the recordings, precise timing information must be preserved throughout data processing.
Contribution

The gr-analysis module introduces several tools that:

- Leverage metadata headers to simplify post-processing of large amounts of data
- Enable synchronized high sample rate recording over extended periods of time (> 30s)
- Tools consist of C code and Python scripts (not GR blocks)
Metadata

- Metadata simply means data about data
- It’s necessary for understanding the meaning and significance of the data itself
  - For example, knowledge of the sampling rate allows mapping from a sequence of samples to a continuous-time waveform
Metadata in GNU Radio

- GNU Radio defines a convention for metadata headers
  - Each piece of metadata is a key-value pair in a dictionary (i.e. “rx_rate” -> “10e6”)
  - Uses polymorphic types (PMT)
  - Could be inline (stored within the same file as data) or detached (stored in external file)
fileman *(for manipulating files)*

- Truncates data files to slices based on time or sample number (provided in headers)
- Interpolates timestamp information between headers as necessary
- Converts between data formats (fc32 and sc16)
  - For hardware reasons, may want to record data as sc16, but most tools prefer data as fc32
fileman usage

- Ex: A data recording (fc32 format) for 5 mins at 25 Msps is around 60 GB.
  - Convenient to take smaller slices of the file to process individually
  - Can pull it into MATLAB for additional processing

- Callable from Python
mkheader (to make GR headers)

- Some of our data was recorded without GNU Radio metadata headers
- mkheader allows generation of headers after the data has been recorded, in order to take advantage of other tools which leverage metadata
- Can also generate “fake” headers for other programs which require headers to operate (i.e. fileman)
- Callable from Python
metadata_to_csv

- Converts metadata from GNU Radio header format into a CSV table of metadata values
- Allows for easy import of metadata into MATLAB/Octave
- Can specify extra key-value pairs (in addition to the standard ones) which are needed for CSV output
High sample rate capture

- Over extended periods of time, the GNU Radio File Meta Sink block fails to perform well at high sampling rates.
- Even without metadata recording, the built-in tools for capturing data fall behind at high sample rates.
specrec

- gr-analysis’s specrec tool is capable of recording data and metadata at much higher rates.
- Uses producer-consumer model to take advantage of concurrency
- Also uses Linux-specific sync_file_range call to force writing page-aligned data
High sample rate capture performance

Capabilities of uhd_rx_cfile and specrec while recording data and metadata at various rates:

<table>
<thead>
<tr>
<th>Sample rate (Msps)</th>
<th>uhd_rx_cfile</th>
<th>specrec</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>10</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>15</td>
<td>*</td>
<td>✔</td>
</tr>
<tr>
<td>20</td>
<td>✘</td>
<td>✔</td>
</tr>
<tr>
<td>25</td>
<td>✘</td>
<td>✔</td>
</tr>
<tr>
<td>30</td>
<td>✘</td>
<td>*</td>
</tr>
</tbody>
</table>

* Works without metadata
Performance comparison

Recording at 20 Msp (iostat data for each second)

- **uhd_rx_cfile**
  - Mean: 67.3
  - Var: 2.08e4

- **specrec**
  - Mean: 69.5
  - Var: 5.79e2
Storage media specifications are often unsustainable over long periods of time. Maximum write speed may be less than advertised.

- `uhd_rx_cfile` writes in bursts, which may exceed storage specifications, leading to overflows.

- `specrec` writes continuously, with a lower maximum rate than `uhd_rx_cfile` demands
specrec limitations

- Not a GNU Radio block (yet)
- Less rigorous, but faster, metadata writing
  - Does not respond to changes in input stream because specrec does not re-tune, etc.
  - Does not re-write any existing headers
  - Does not write inline headers
Available today!

- Our gr-analysis module is now available
- GitHub: https://github.com/garverp/gr-analysis
Thank you!
Metadata in GNU Radio

- Each header is expected to contain these items:
  - version: (char) version number (usually set to METADATA_VERSION)
  - rx_rate: (double) Stream's sample rate
  - rx_time: (pmt::pmt_t pair - (uint64_t, double)) Time stamp (format from UHD)
  - size: (int) item size in bytes - reflects vector length if any.
  - type: (int) data type (enum below)
  - cplx: (bool) true if data is complex
  - strt: (uint64_t) start of data relative to current header
  - bytes: (uint64_t) size of following data segment in bytes

- Extra key-value pairs also allowed