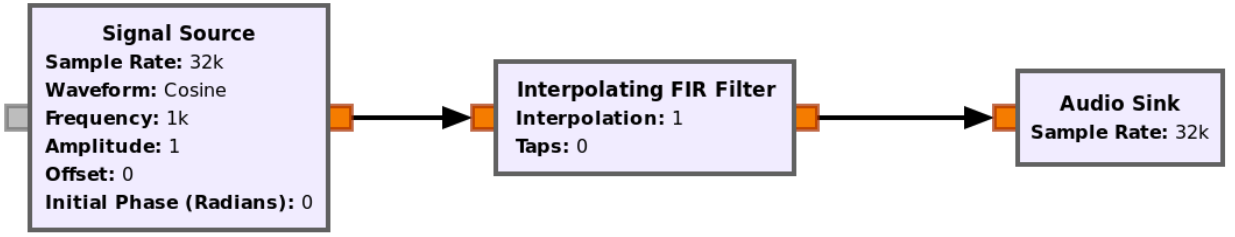
Managing Latency in Continuous GNU Radio Flowgraphs

Matt Ettus
matt@ettus.net
GNU Radio Conference 2019

Problem Statement

- What is Latency and where does it come from?
  - Inherent
  - Scheduler induced
    - Granularity of calls to work function
    - Thread latency, scheduler overhead, etc.
  - **Buffer induced**
- What causes buffer-induced latency?
- When is it a problem?
  - Generally only on TX
  - The „SkyBox Problem“
  - See audio_latency.grc flowgraph

Buffers and the Scheduler



- What do the arrows really represent?
  - What happens when the arrows split?
- How does the scheduler work?
- What controls the rate of computation
- Great article by Marcus
  - https://www.gnuradio.org/blog/2017-01-05-buffers/

Solutions

- Shrink buffers?
  - myblock.set_max_output_buffer(num_items)
  - NOOOOOO!!!!!!
    - Even minimum sized buffers can be too big in a large flowgraph
    - Some computations require buffers of a certain size
  - But *may* be useful to control scheduler-induced latency
- Drop items already in flight?
  - Dangerous, creates discontinuities
- Intelligently control the filling of buffers
  - Active Latency Management
  - Limit the number of in-flight data items between decision point and consumption
  - See solution flowgraph

latency_manager

```
int
latency_manager_impl::work(int noutput_items,
    gr_vector_const_void_star &input_items,
    gr_vector_void_star &output_items)
{
  const char *in = (const char*) input_items[0];
  char *out = (char *) output_items[0];

  int copy_count = std::min(noutput_items, d_tag_phase + d_tokens * d_tag_interval);
  std::memcpy(out, in, copy_count * d_itemsize);

  int tag_loc = d_tag_phase;
  while (tag_loc < copy_count) {
    d_tag.offset = nitems_written(0) + tag_loc;
    d_tag.value = pmt::from_long(tag_loc);
    add_item_tag(0,d_tag);
    tag_loc += d_tag_interval;
    d_tokens--;
  }
  d_tag_phase = tag_loc - copy_count;
  if(copy_count == 0) {
    boost::this_thread::sleep(boost::posix_time::microseconds(long(100)));
  }
  return copy_count;
}
```

Use Cases

- Continuous Transmissions
  - Interactive Signal Generator
  - CW, RTTY, PSK31 for Hams
  - Dead man switch / self-destruct
  - Satellite TT&C (telemetry, control, and alarm) links
  - Muxed satellite downlink
- True Priority Muxes
- Realtime feedback control systems, hardware in the loop
- Can be extended for:
  - Mixed sample rate flowgraphs, and the two-clock problem
    - Control the arbitrary resampler to ensure buffer levels
  - Graceful handling of underruns

Limitations and Future Work

- Only buffers within the feedback loop are managed
  - Sources could implement the latency manager
  - Sinks should report the latency tags
- Hardware buffers should be included in the loop as well
  - audio_sink
  - uhd_sink

- **Thanks again to Derek**
- **Find the code at https://github.com/dkozel/gr-workinprogress**
- **Email -- matt@ettus.net**
- **Twitter -- @Matt_Rambling**
- **I’m hiring (@ Apple)**
  - **Communication Systems and Software Engineering**
  - **Intern (summer, winter, year-long)**
  - **Full Time**