Background

In the testing for the *fddaq-v4.4.3* patch release, Wes noticed that the PDS part of the `readout_type_scan` automated integration test in the `daqsystemtest` package was failing.

- “Request timeout” warnings; empty fragments

This was triggered by the simple change of the number of DAPHNE frames per superchunk from 12 to 1 (link).

- This is in DAPHNE data, not DAPHNEStream data, so it only affects the internally-triggered PDS readout mode.

No problems were observed in the tests done with real electronics.

Investigation

In looking into the behavior of the integtest system with 12 frames-per-superchunk vs. 1 frame-per-superchunk, I noticed that the timestamp difference between subsequent PDS data frames (in `DAPHNEFrameProcessor`) went from small (~25 usec) to really, really small (< 1 usec).

- Recall that many of our integtests use emulated data, in which data is read from reference files, timestamps are updated, etc.

I believe that the very slowly changing timestamp values in the ‘1 frame per superchunk’ scenario is destructively interfering with idiosyncrasies in the `TimestampEstimator` to cause non-trivial mismatches between the trigger times (and readout window start/end times) and the timestamps of the data stored in the latency buffer.

Idiosyncrasies of the TimestampEstimator

There are lots... The two that I'll mention today are the following:

1. It tries to account for delays in the transport of `TimeSync` messages by comparing the Linux system times (wallclock times) when each message was sent and received, and it uses that difference to correct the DUNE-time timestamps in the messages.
   1. I hope that you can see the problem with combining two times that are changing at possibly very different rates…
2. There is a bug in our system configuration scripts that has the `FakeHSI` sending `TimeSync` messages to itself.

Taken together, these two features can cause `DataRequest` times to become progressively more out-of-step with the data times that are in the latency buffers.

Mea culpa

Regarding these two idiosyncrasies of the `TimestampEstimator`…

I/we should have fixed or removed the message-transport-delay logic long ago, but I wanted to fully understand how it would affect various uses of the `TimestampEstimator` first, and I haven't yet made time for that.

I should have fixed the sending of `TimeSync` messages from the `FakeHSI` long, long ago…

What should we do for v4.4.4?

Anything?

Three small changes that help to get things working again:

1. Increase the `daphne_time_tick_diff` in `FDFakeCardReader`
    1. Candidate change is here
    2. It would be great to learn what value this variable should have, based on knowledge of the electronics
2. Ignore `TimeSync` messages from oneself
    1. Candidate change is here
    2. This isn't as elegant as turning off these `TimeSync` messages in `daqconf`, but maybe it's fine for now
3. Update the expected fragment sizes for PDS fragments in the `readout_type_scan` in response to change (1)

Other integtest notes

[Recall that one of the tricks that we play in the integtests is to use a WIBEth data file that has all zeros for the ADC values coupled with features in the `SourceEmulatorModel` class (e.g. here) that set a configurable fraction of the ADC values to large numbers so that they result in TPs being created. In this way, we can tune the rate of TPs found in an emulated-data system.]

When our typical system configurations were changed to use one TP `DataRequestHandler` per APA plane (instead of one per all three planes), I noticed that there were occasional request-timeout warnings at the start of runs when TPG was enabled.

I believe that this is simply because the random setting of high ADC values in the emulated data, and the resulting generation of TPs, needs a little time to produce TPs for all three planes. Adding a 2-second delay between the start of a run and the enabling of triggers helped, so I have made that change in several integtest.