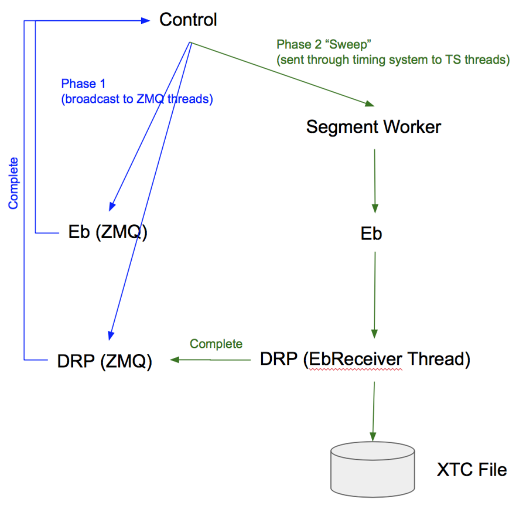
Two Phase Transitions

We propose to use a two-phase approach for each transition (after CONNECT) in the DRP, inspired by the LCLS-I approach

- The first phase is handled by a ZMQ broadcast, so configures can happen in parallel
- The second phase is handled in the timing system thread. This phase "sweeps" out the results from the first phase
- The control level sends out the second phase after the first phase is completed
- We will try to run as much code as possible in the ZMQ thread in order to make the TS thread "sweep" as quick as possible
- The timing system thread is responsible for all xtc writing
- If a DRP has N segment-level workers, only one of them will receive the timing system transition
- Since the mon nodes quickly cache the relevant transition, their "completion" is ignored in this process
- All timeouts for the two phases are done by the control level
    - each node's first-phase transition (maybe just configure and configUpdate) specifies a timeout value, perhaps with the CONNECT collection message
    - hopefully the second phase doesn't need a transition-dependent timeout, but if it does it will be specified in a similar manner to the first phase
- The ZMQ thread should inform the timing-system thread of its config JSON, so it can be appended to the XTC
- The timing-system thread's "complete" message is transmitted via the ZMQ thread, since that thread has knowledge of the appropriate sockets.

Some implementation details:

- I think this is done with the "inprocSend" ZMQ context in DrpBase.cc.
- The phase1 response to the control level from the drp nodes is in PGPDetectorApp.cc:handlePhase1()



MEB Discussion

April 15, 2022: claus, caf, cpo

Ric found that in UED the disable transitions were being delayed by several seconds, queueing up a few of them and creating buffering problems for the meb and difficult-to-understand crashes (perhaps because we only have 1 buffer for the disable transition?). We discussed two options to address this, allowing the meb to participate in the control.py decision about when to execute the next transition:

option (1) is having the meb participate in the phase2 sweep (like teb)
  - more work for ric
  - have to generate the "inproc" (complete) message
  - complication: has to handle slowupdate in a special way
  - more self-contained
  - ric worries that meb buffers may not be promptly returned to the drp: maybe wouldn't work?

option (2) is meb becomes like a drp: generate it's on phase2 complete and send to control.py via ZMQ "inproc" message
  - more work for caf
  - could more precisely identify the meb as being a problem if meb crashed
  - touches both drp and control.py code

does the above decision affect speed of phase2?
- i think the answer is no: meb doesn't do anything in phase2

tentative decision is to try option (2)