Towards a PAR (or PARs) for Pulsed Queues

Norman Finn
Huawei Technologies Co. Ltd
nfinn@nfinnconsulting.com
new-finn-pulsed-queuing-0821-v03

Purpose of this presentation

- The ultimate goal of this presentation is to generate a PAR for a new transmission selection system that will provide the absolute guarantees for latency and congestion loss now offered by Asynchronous Transmission Selection, using no per-Stream state in the interior of the network, and with a minimum of features in addition to IEEE Std 802.1Qav FQTSS (output scheduling).
- This means that Streams can be created and deleted without changing the state of interior network nodes, either by protocol or by configuration.
- The end-to-end latency will not be as good as for ATS, but the author believes that the flexibility makes this idea very attractive.

Purpose of this presentation

- Mick Seaman has described the Paternoster shaping algorithm, most recently in cr-seaman-paternoster-policing-scheduling-0519-v04.
- I have described Multi-level Cyclic Queuing and Forwarding (Multi-CQF) in df-finn-multiple-CQF-0919-v02 and the accompanying df-finn-multiple-CQF-slides-0919-v01.
- I will show in this presentation that these are examples of “Pulsed Queuing”, and that they are useful in combination.
- I will make a case to create one or more PARs to standardize them.

Syntonization vs. Synchronization

Correcting the confusion

- Johannes Specht, in new-specht-non-fifo-queues-0721-v01, seized on an unfortunate choice of words in my df-finn-multiple-CQF-0919-v02 and came to a very understandable, but wrong, conclusion about Multi-CQF.
- The confusion is between synchronization and syntonization requirements for CQF. A careful reading of df-finn-multiple-CQF-0919-v02 will show what I meant in my use of these terms, but the following slides will make this clear.

Requirements for CQF or Multi-CQF

- In the absence of per-Stream reshaping, all of the nodes along the path of a Stream must use either the same CQF cycle time, or a cycle time that increases by an integral multiple of the last hop's cycle.
- All nodes in a CQF network must run at exactly the same frequency, in the sense that the difference in the number of CQF cycles counted by two different nodes must be the same, within a fraction of one cycle, over an arbitrary period of time.

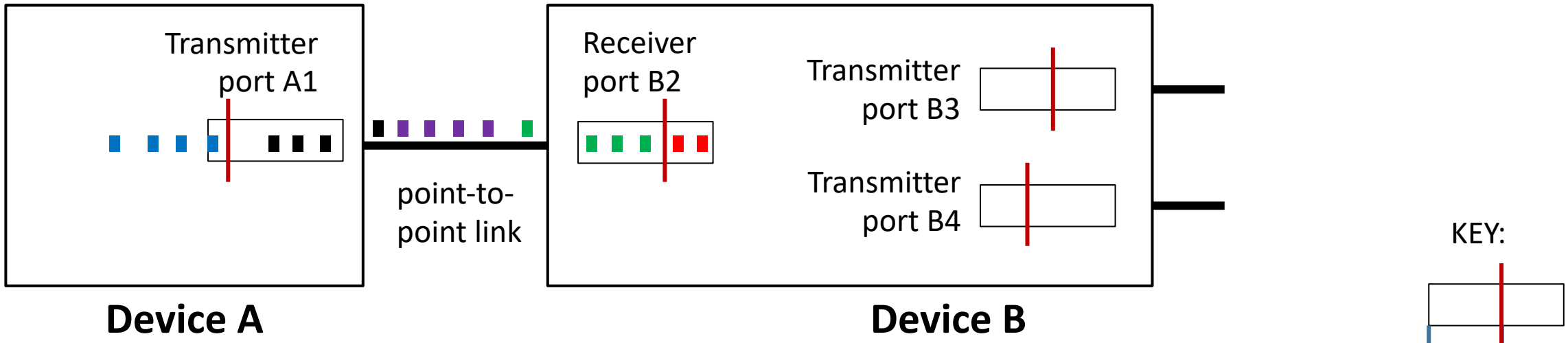
Requirements for CQF or Multi-CQF

- In Annex T of IEEE Std 802.1Q-2018, it is stated that all nodes in a network synchronize both their input and output cycles so that all receive and transmit windows start simultaneously throughout the network. As shown in df-finn-multiple-CQF-0919-v02, this is not necessary.
  - In particular, the input windows, which assign frames to particular buffers, need not start in synchrony with the output buffers. Rather, they must be in synchrony with the previous node's output cycle, offset by the link delay. This means that the input windows are offset in phase from the output windows in the same node.
  - The various output ports of a node need not start their transmit windows in synchrony; they can be offset from each other in phase.

Requirements for CQF or Multi-CQF

- In order for all nodes to transmit simultaneously, Annex T of IEEE Std 802.1Q-2018 requires time synchronization, presumably by PTP.
- However, PTP is not necessary; SyncE (synchronous Ethernet, ITU-T G.8261, G.8262, and G.8264) is sufficient, without synchronizing a value of time among nodes.
- This is because the input node is synchronized with its connected transmitter, offset by the ((link delay) modulo (cycle time)). That is, the absolute link delay is unimportant; only the phase offset of the window times caused by the link delay is important, and this is a **one-way measurement**.
- It is in this sense that syntonization, not synchronization, is needed.

Input / output phasing

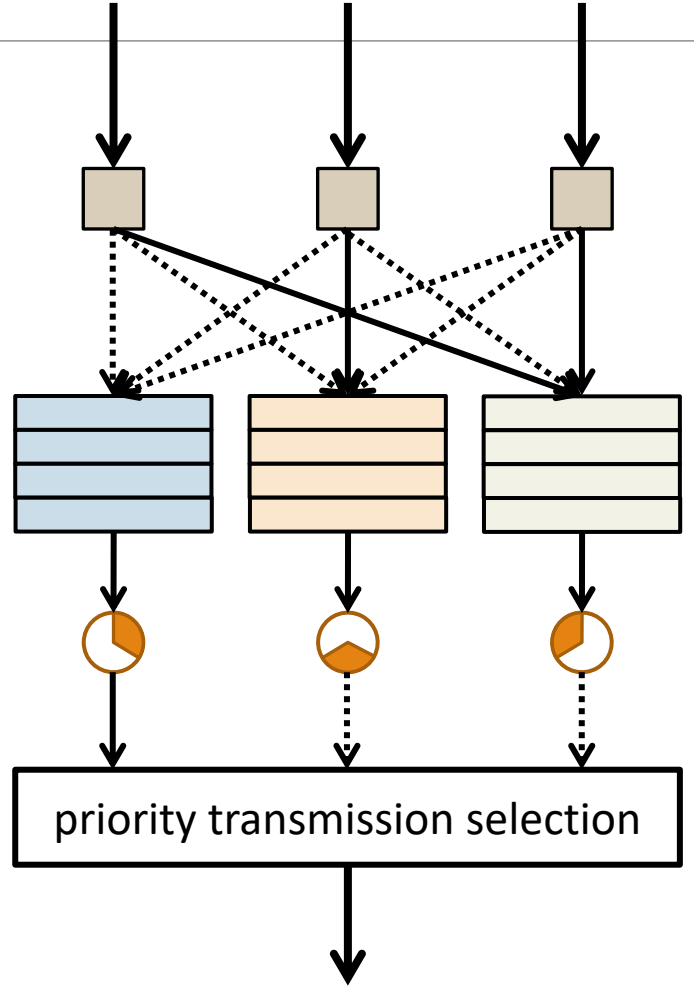


- We can assume that the receiver port B2 can measure the phase differences between itself and the transmission ports B3, B4. (By the nature of CQF, all run at the same window cycle time.)
- Receiver port B2 must determine the phase offset between transmitter port A1 and receiver port B2, and remove it.

What are Pulsed Queues?

Current queuing models

- Paternoster and Multi-CQF have been described, so far, as using two or more queues, along with a timer mechanism to enable one queue at a time, in sequence, for transmission selection.
- A different class of service is used for each queue.



input ports or Streams

time- or byte-count-based queue selection

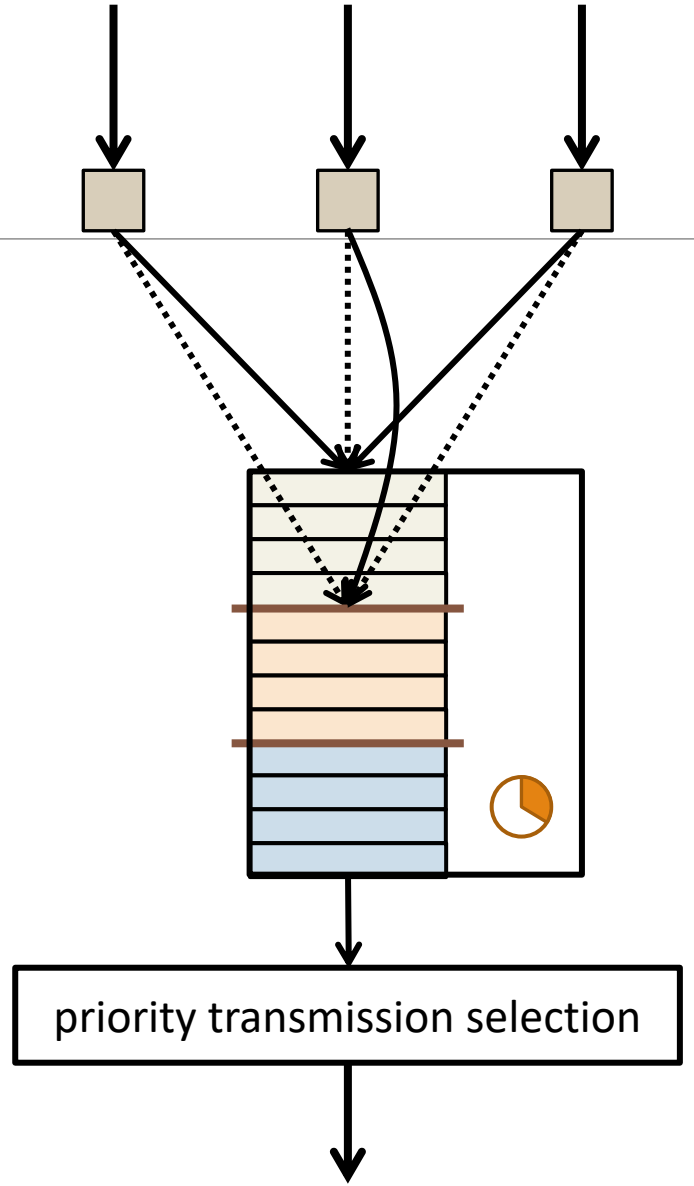
queues

P802.1Qbv timed transmission gates

output port

Pulsed Queuing

- A “Pulsed Queuing” model is preferable, going forward.
- Queue selection in the current model becomes bin selection in the Pulsed Queuing model.
- One class of service queue has multiple bins that are rotated (enabled for transmission selection) by an internal clock.



input ports or Streams

time- or byte-count-based **bin** selection

multiple bins in one queue

time-based bin rotation

output port

Why a new model?

- The current model for P802.1Qch CQF was selected because:
  1. Timed output gates (P802.1Qbv) are needed for time-division multiplexing and for accurate delivery times.
  2. Timed input filters (P802.1Qci) are needed for defense against errors.
  3. The current CQF model **minimized the total word count** in P802.1Qch (CQF) + P802.1Qci (input filters).
- One time constant per port cannot offer the range of services required by a provider; multiple time constants per output port are required. This takes too many queues in the current model.
- We have become more comfortable, via P802.1Qcr, with a more complex model for a class-of-service queue than a simple FIFO.

One model: Output timing

- Paternoster and Multi-CQF rotate buffer roles in the same manner, based on a clock.
- Both methods can run at different frequencies for different classes of service on the same port.
- The difference is that the Multi-CQF buffer rotation clock is synched with all nodes in the network, while the Paternoster rotation clock is not synched, but runs at the same frequency as other nodes (within some tolerance).
- It is trivial to roll buffer transmission rotation into a single model.

Two variants: Input bin selection

- In both Paternoster and Multi-CQF, each frame is assigned to a queue (now, a bin).
- Two variants are required for bin selection:
  1. Multi-CQF bin selection is performed using purely time-based input filters run by a synchronized clock. Filters are per-class of service.
  2. Paternoster bin selection is performed by per-Stream, per-output port byte counters.

  (Depending on complexity/capability tradeoffs made, other per-'s are possible.)
- A single model can be used for conveying the bin selection result to the output queue.

Why bring up Paternoster?

- At the edge of the network, a Sender may not use CQF.
- If it does not use CQF, then defining the Multi-CQF parameters to carry that Stream (cycle size and bits-per-cycle) based on the Stream Reservation Protocol parameters (max frame size, max frames per measurement time) leads to gross over-provisioning. (See dd-finn-CQF-and-shaping-0120-v01.)
- Paternoster meshes very well with Multi-CQF at the edge of the network, allowing a little extra buffering at the edge, and minimizing over-provisioning. For example, Paternoster can be used on an input port that feeds the same output buffer as another input port that uses CQF timing.

One document

- One document (tentatively called “Pulsed Queuing”), can thus combine the notions of:
  - Multiple bins per class of service queue, multiple queues per output port.
  - Rotating roles for the bins (transmitting, holding, collecting, etc.) based on a clock whose period depends on the class of service.
  - Bin selection by a filter state machine.
- And that document can have separate sections describing the bin selection filters for:
  - Paternoster, and
  - Multi-CQF.

What is *not* suggested for inclusion in a near-term PAR

Deadline scheduling, timing information included in frames, …

- There are may schemes (some mentioned in new-specht-non-fifo-queues-0721-v01) that use insertion sort techniques to place a frame in a particular place in an output queue, based on some calculation performed on the contents of a frame.
- In addition, schemes have been suggested in IETF for marking packets with the information that varies from cycle to cycle, so that information in the packet can be used to select the output buffer, without a sort operation.
- Some discussion of these and similar ideas is necessary, of course.

Who wants Pulsed Queues?

Service Provider vs. industrial networks

- Service Providers reviewing TSN in general, and P802.1DF (TSN for Service Providers) in particular, have expressed concern over the differences between their environment and that of industrial nets:
  - An SP serves more customers; more Streams are needed.
  - An SP's Streams must be created and deleted dynamically, without interference to existing Streams.
  - An SP needs to offer a range of cost/benefit values for classes of service.
  - A Stream can traverse a wide range of link speeds over its path.

Bundling

- The differences on the previous page lead to a requirement for **bundling**.
- Many Streams have to be bundled into a single Stream.
  - This saves per-hop state machines.
  - This saves management activity.
  - This saves buffer space and improves latency. (100 identical streams require a 100-packet bin!)
- Bundle joining and merging is required.
- This may be a separate PAR.

Multi-CQF, Paternoster, and Bundling

- For providers to whom synchronization is a good tradeoff against per-Stream state machines, Multi-CQF is useful.
- But, Paternoster is necessary to re-distribute the Streams' frames into bins:
  - At ingress to Multi-CQF;
  - At points where the bin rotation timer period changes; and
  - At every point where bundles of Streams are split or merged.
- **Paternoster is a relatively low-cost tool for bundling, and necessary to make Multi-CQF work well, at least at the edges.**

The (dis-)advantages of PQ for the SP

- The per-Stream state machines for Paternoster are simpler than those for ATS. Multi-CQS requires no per-Stream state machines.
- The worst-case latency computation is vastly simpler for the PQ schemes than for other methods, because adding or deleting a Stream does not affect any other Stream's latency.
- ATS yields somewhat lower latency and higher bandwidth utilization than PQ.
- Multi-CQS requires at least syntonization.
- But, of course, **all three schemes can be used in one Bridge**. It's all a matter of optimizing costs.

Improvements that can be included

- The cited Paternoster and Multi-CQF descriptions both offer additional improvements to the basic algorithms that are, for the most part, applicable to both mechanisms, including:
  - Preemption of slower-cycled PQ queues increases the maximum bandwidth available to faster-cycled queues.
  - Sausage making (splitting/combining the various-sized frames of a single Stream [or bundle of Streams] into uniformly-sized frames) reduces the overprovisioning for all PQ classes of service.
  - Overprovisioning a Stream, by assigning it a class of service with a faster cycle time than its bandwidth requires, can reduce its latency.

Proposal

Conclusion

- I propose that we work towards a PAR for Pulsed Queues for approval at an upcoming IEEE 802 plenary.
- It will address, at least, Paternoster and Multi-CQF,.
- We can talk more about what additional features, if any, should be included.

DISCUSSION

Thank you