Towards a PAR (or PARs) for Pulsed Queues

Norman Finn
Huawei Technologies Co. Ltd
nfinn@nfinnconsulting.com
new-finn-pulsed-queuing-0121-v02

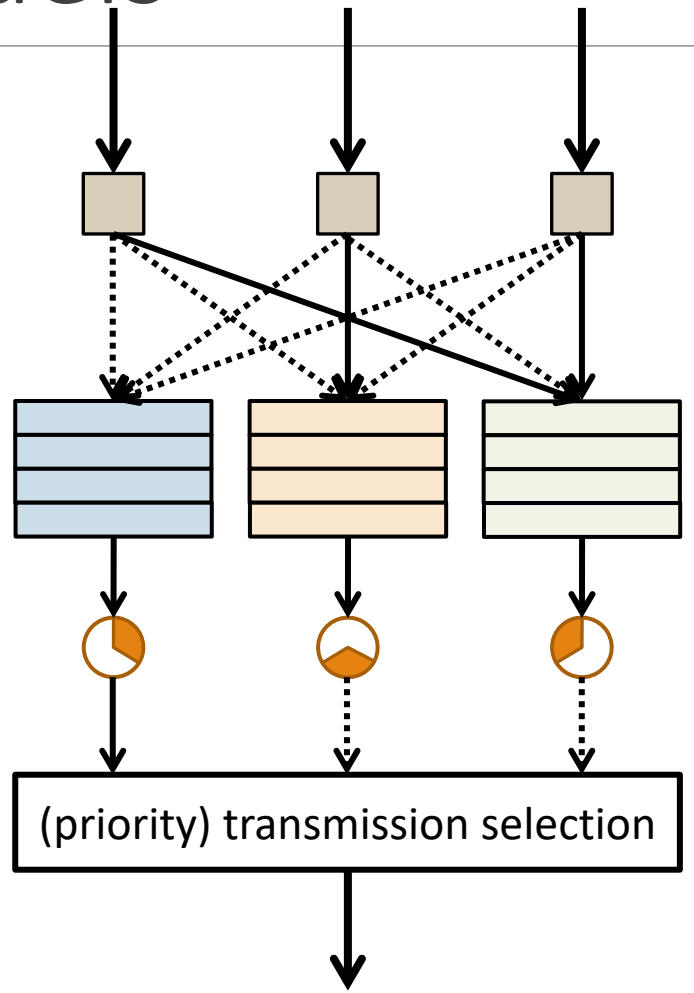
Purpose of this presentation

- 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
  1. Two examples of “Pulsed Queuing”; and
  2. Useful, especially to service providers.
- I will make a case to create one or more PARs to standardize them.

What are Pulsed Queues?

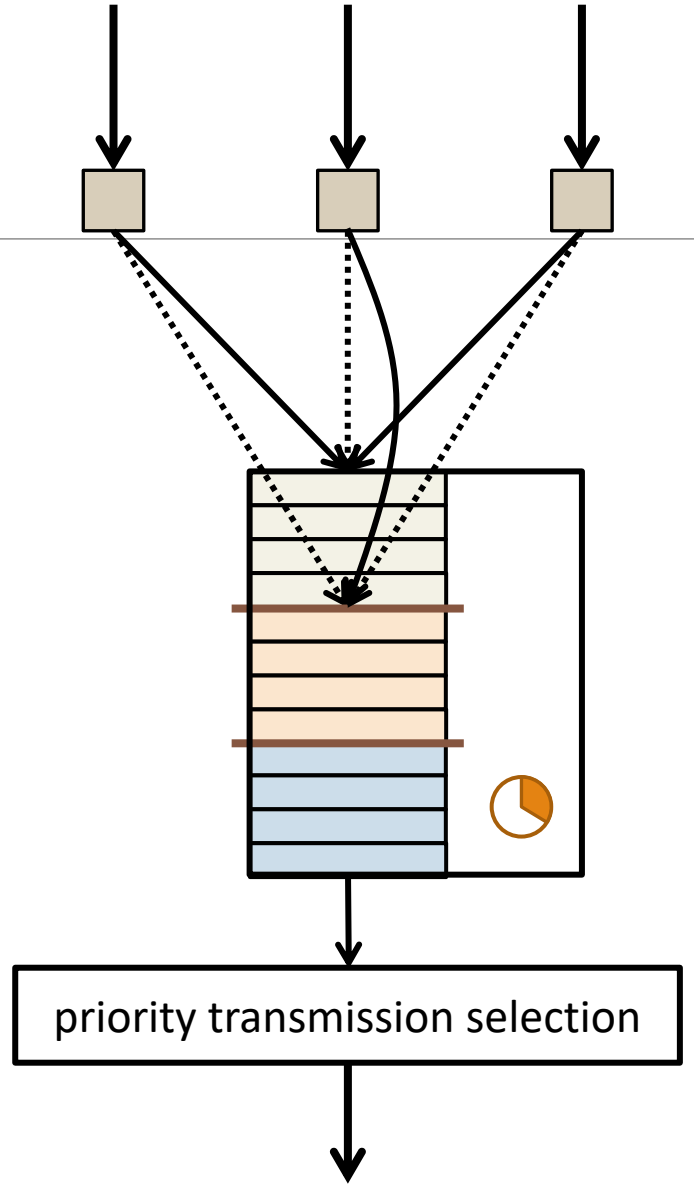
Current queuing models

- 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.



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

CQF time-based or Paternoster 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) were needed for time-division multiplexing and for accurate delivery times.
  2. Timed input filters (P802.1Qci) were needed for defense against errors.
  3. **The current 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.

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.
  - Regular rotating roles for the bins (transmitting, holding, collecting, etc.) based on a clock whose period depends only on the class of service.
  - Bin selection by a filtering state machine running on that same clock.
- And that document can have separate sections describing the bin selection filters for:
  - Paternoster and
  - Multi-CQF.

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 frequently, 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 along 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.
  - Bundling saves per-hop state machines.
  - Bundling saves management activity.
  - Bundling saves buffer space and improves latency. (100 identical streams require a large, slow bin ≥ 100 frames. Bundling them into one stream allows 1 frame per bin and 100 times faster rotation period.)
- Bundle joining and merging is generally required inside the network.

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 speeds up, which is precisely when it is most useful to bundle Streams together.
- **Paternoster is a relatively low-cost tool for bundling, and necessary at the edges to make Multi-CQF work.**

The (dis-)advantages of PQ for the SP

- The state machines for Paternoster are simpler than those for ATS. Multi-CQS requires no per-Stream state machines.
- The worst-case latency computation is 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 lower latency and higher bandwidth utilization than PQ.
- Multi-CQS requires at least syntonization.
- But, of course, **all three schemes can be used on 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:
  - Preempting 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, reduces its latency.
  - Providing extra bins to perform latency matching on diverse 802.1CB paths.

Proposal

Proposal

- I propose that we work towards a PAR for Pulsed Queues for approval at the July IEEE 802 plenary.
- It will address, at least, Paternoster, Multi-CQF, and bundling.
- We can talk in March and/or May about what additional features, if any, should be included.

DISCUSSION

Thank you