# Towards a PAR (or PARs) for Pulsed Queues

Norman Finn
Huawei Technologies Co. Ltd
nfinn@nfinnconsulting.com
new-finn-pulsed-queuing-0821-v04

## Why?

# Why a new PAR and document?

- Asynchronous Transmission Selection (ATS, IEEE Std 802.1Qcr) provides bounded end-to-end delivery time and freedom from congestion loss, but apparently requires per-stream state machines to be configured at each hop along the path, and considerable computation effort when a stream is added to the network.
- The Credit Based Shaper (CBS) can also, with sufficient analysis, provide bounded delivery time and zero congestion, again with per-hop configuration at stream add time.
- Cyclic Queuing and Forwarding (CQF, IEEE Std 802.1Q-2018 Annex T) provides the same guarantees, with trivial calculations and no per-hop configuration when adding a stream, but with much longer delivery times, and such poor efficiency, scalability, and flexibility that **CQF is not useful in its present state.**
- We will show that **Pulsed Queuing, Multi-CQF and Paternoster are (mostly) special cases and/or optimizations of the more general Asynchronous Traffic Shaping.**

## Detailed descriptions

- Mick Seaman has described the Paternoster shaping algorithm in cr-seaman-paternoster-policing-scheduling-0519-v04.
- Norman Finn has described Multi-level Cyclic Queuing and Forwarding (Multi-CQF) in new-finn-multiple-CQF-0921-v02, and a method for synchronizing a CQF receiver with the transmitter from which it is receiving, in new-finn-CQF-sync-method-1121-v1.

## What is Pulsed Queueing?

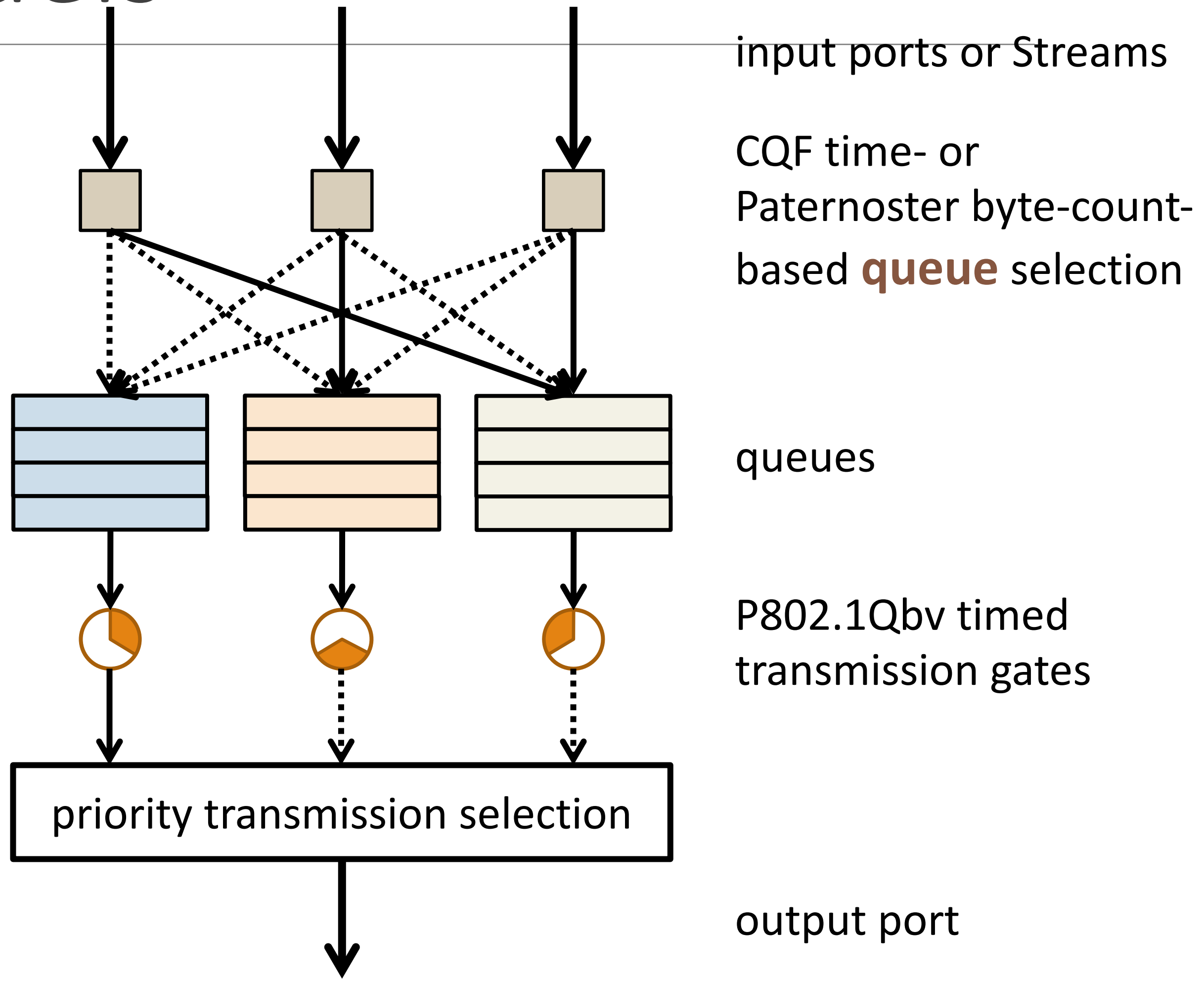
## What is Pulsed Queuing?

Pulsed Queuing has **five parts**:

1. A transmission selection algorithm, **Pulsed Queuing**, that divides a class-of-service queue into some number of logical bins that are output in strict rotation at a constant frequency;
2. An algorithm, **multi-Cyclic Queuing and Forwarding (mCQF)**, for storing received frames into Pulsed Queue bins based on the time of reception of the frame;
3. An algorithm, **Paternoster**,  for storing received frames into Pulsed Queue bins based on per-Stream bit/byte counters;
4. An algorithm, **PQ Aggregation**, for aggregating, disaggregating and reaggregating streams into and out of compound streams that is suitable for use with Pulsed Queuing;
5. A protocol for locking a CQF receiving to its transmitter; and
6. A method for perfect **syntonization** of a region in the network, i.e. a means for bounding the cumulative difference in elapsed time as measured between any pair of nodes in the region over an arbitrary length of time.

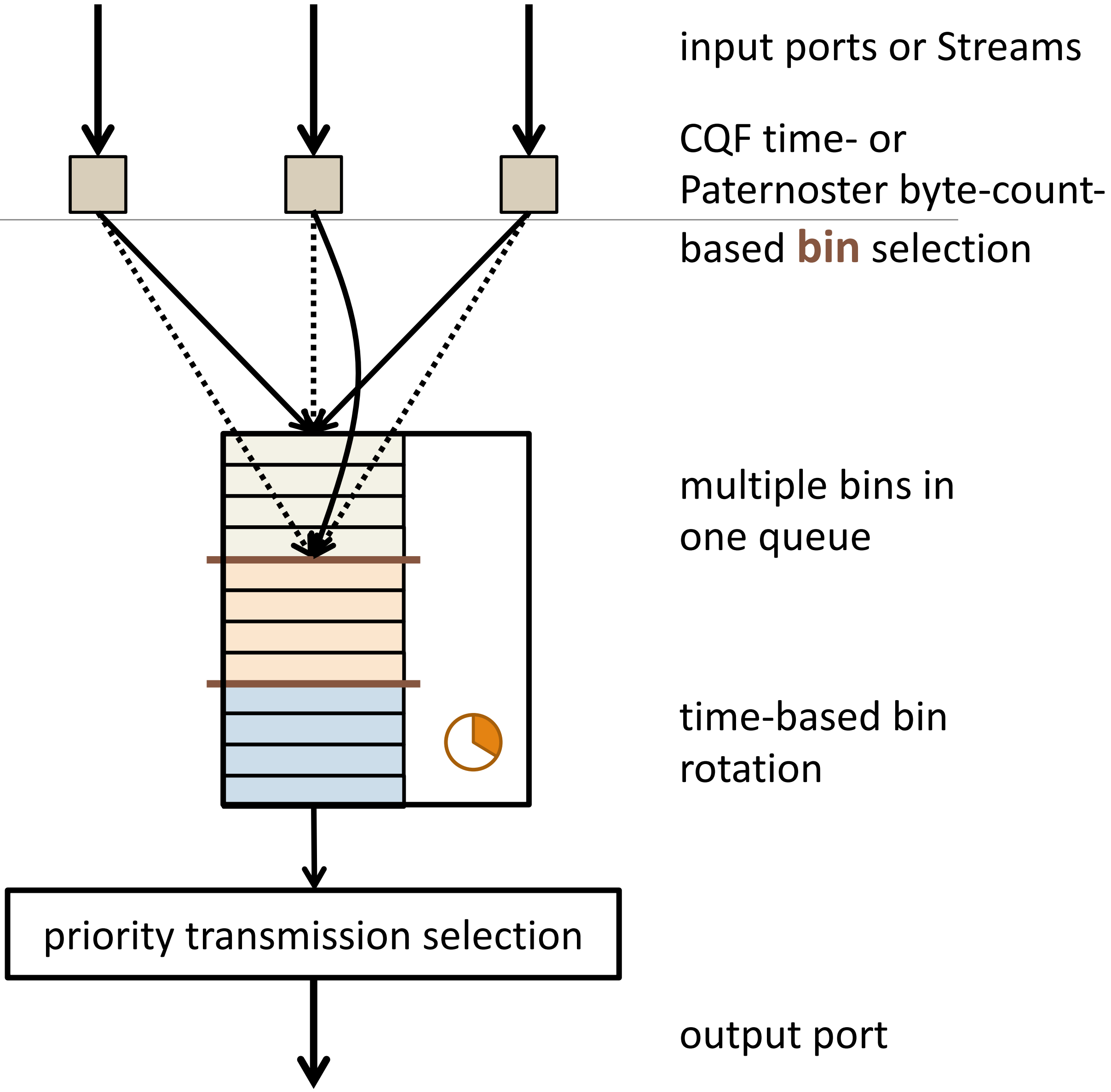
## Current queuing models

- 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
CQF time- or Paternoster byte-count-based **queue** selection
queues
P802.1Qbv timed transmission gates
priority transmission selection
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
CQF time- or Paternoster byte-count-based **bin** selection
multiple bins in one queue
time-based bin rotation
priority transmission selection
output port

## Pulsed Queuing is similar to ATS.

- Pulsed Queuing “bins” are equivalent to assigning a number of frames the same ATS transmission eligibility time.
- The methods described here for assigning frames to a bin (mCQF and Paternoster) differ in detail from those described for ATS.

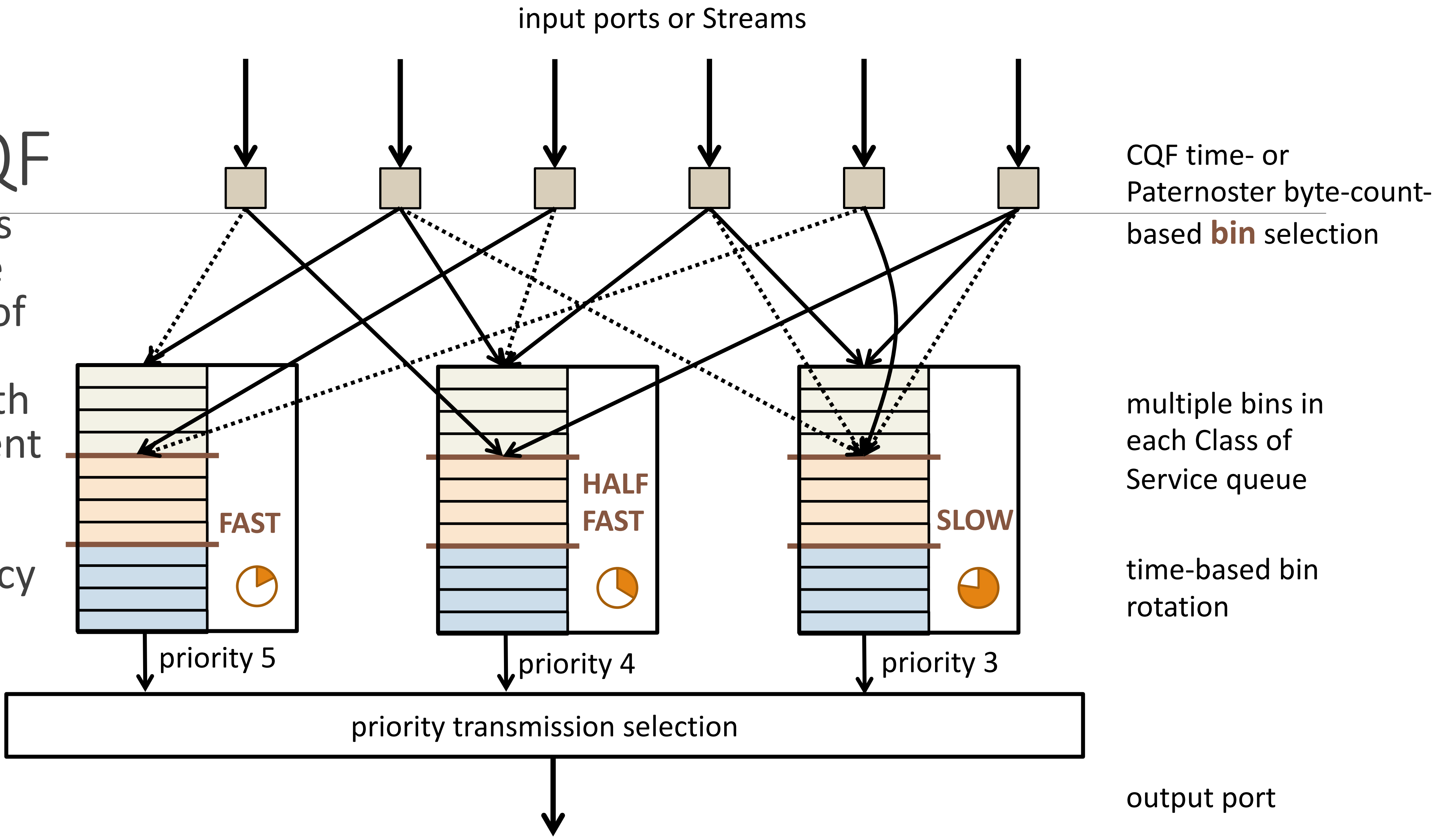
# Multi-Cyclic Queuing and Forwarding and Paternoster

## CQF problems: Inflexibility

- Each CQF buffer is one classes of service queue, so that Scheduled Transmissions (802.1Qbv) can be used to rotate between them. Double buffering (one filling, one draining) takes two Class of Service queues.  This makes more than one rotation frequency impractical.
- But, one rotation frequency is inflexible; a low frequency is required to support many slow streams, but this results in large buffers and long per-hop latency for a fast streams.
- Streams with different bandwidth requirements have different requirements for buffer rotation frequency.

## mCQF

supports multiple classes of service, each with a different buffer rotation frequency

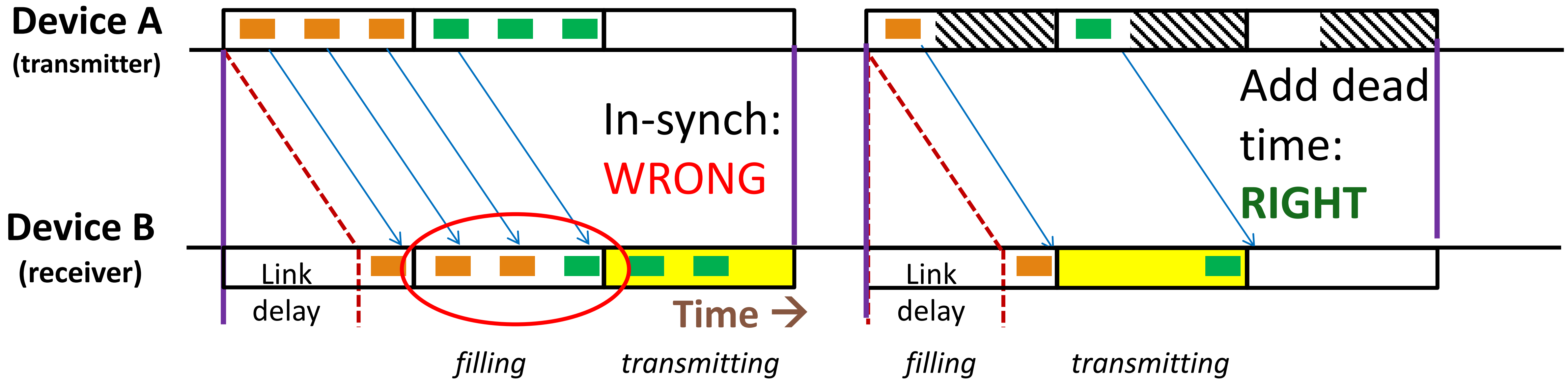


input ports or Streams
CQF time- or Paternoster byte-count-based **bin** selection
multiple bins in each Class of Service queue
time-based bin rotation
FAST
HALF FAST
SLOW
priority 5
priority 4
priority 3
priority transmission selection
output port

# CQF problems: Link delay

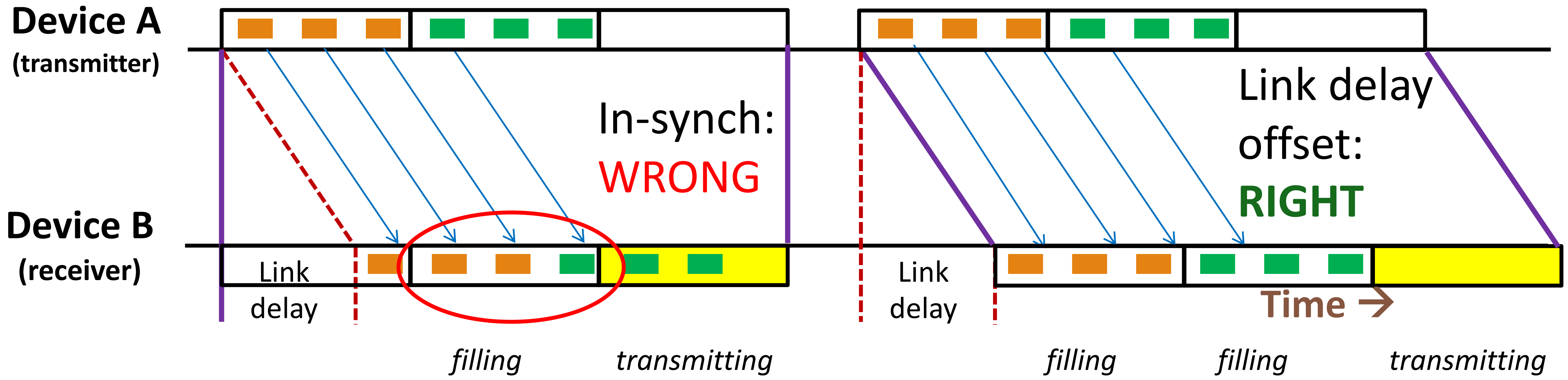
- With two buffers at each hop, one filling and one draining, link delay, worst-case forwarding delay, and all timing uncertainties must be subtracted from the buffer rotation time, reducing the time available for data and/or reducing the buffer rotation frequency.  For many applications (e.g. service providers), this becomes untenable.
- By using more than two buffers, and divorcing the receive time boundaries from the transmit time boundaries, the link delay and forwarding delay no longer directly affect the buffer rotation frequency.

# CQF problems: Link delay: Two buffers



- Annex T of IEEE 802.1Q-2018 fixes long link delay by adding dead time, which reduces the bandwidth available for CQF streams and sets a minimum cycle time.

## CQF problems: Link delay: Three buffers

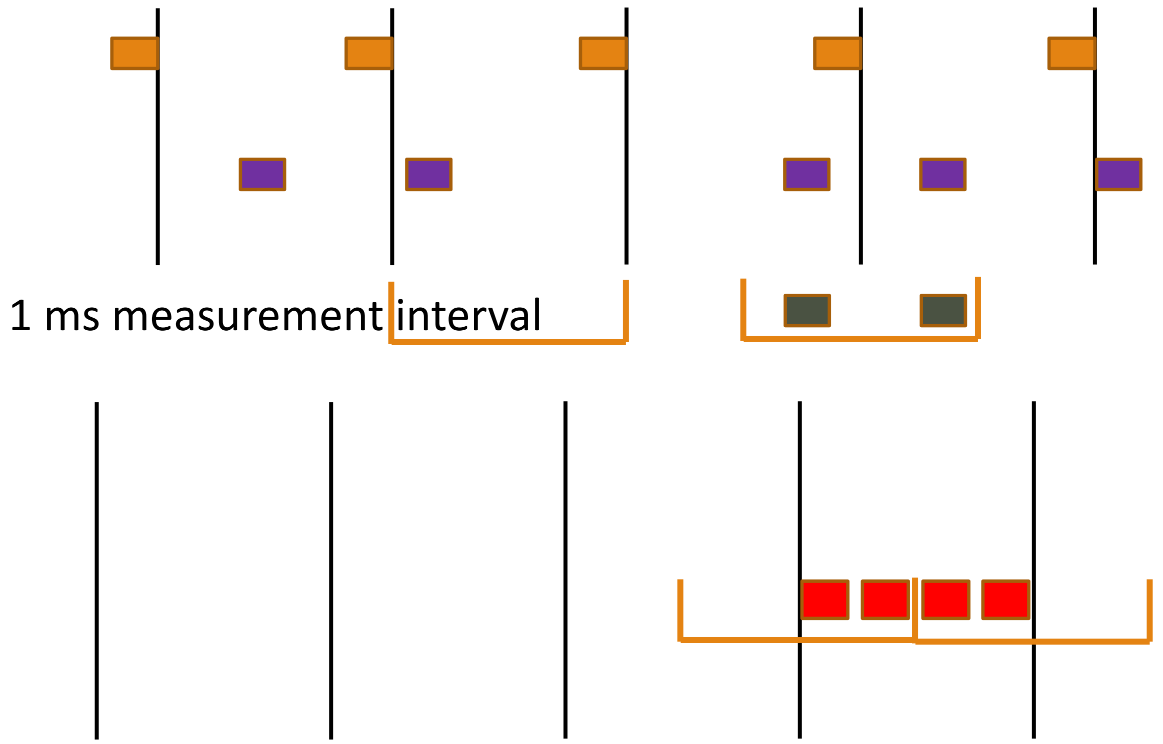


- Offsetting the input clock by the link delay means that link delay does not add to the required dead time. More bandwidth is available for CQF.
- Now, three buffers are required.

## CQF problems: Ingress conditioning

- A stream reservation carried by CQF is implemented by reserving a maximum number of wire-time bits to be transmitted per bin cycle time for the stream.
- But, MSRP and RAP, the stream reservation protocols, define reservations in terms of worst-case number of maximum-size frames in a sliding window (MSRP) or as a bandwidth and worst-case burst size (RAP), which are appropriate for describing non-CQF talkers. However, this requires gross over-provisioning when CQF, as described in Annex T of 802.1Q-2018, is employed.
- Using the Paternoster algorithm for ingress conditioning allows the bandwidth/burst size characterization, and thus non-CQF talkers, to be used efficiently with Pulsed Queuing.

# CQF problems: Ingress conditioning



Stream generated by application: 1 frame per ms

As output by Talker, with interference

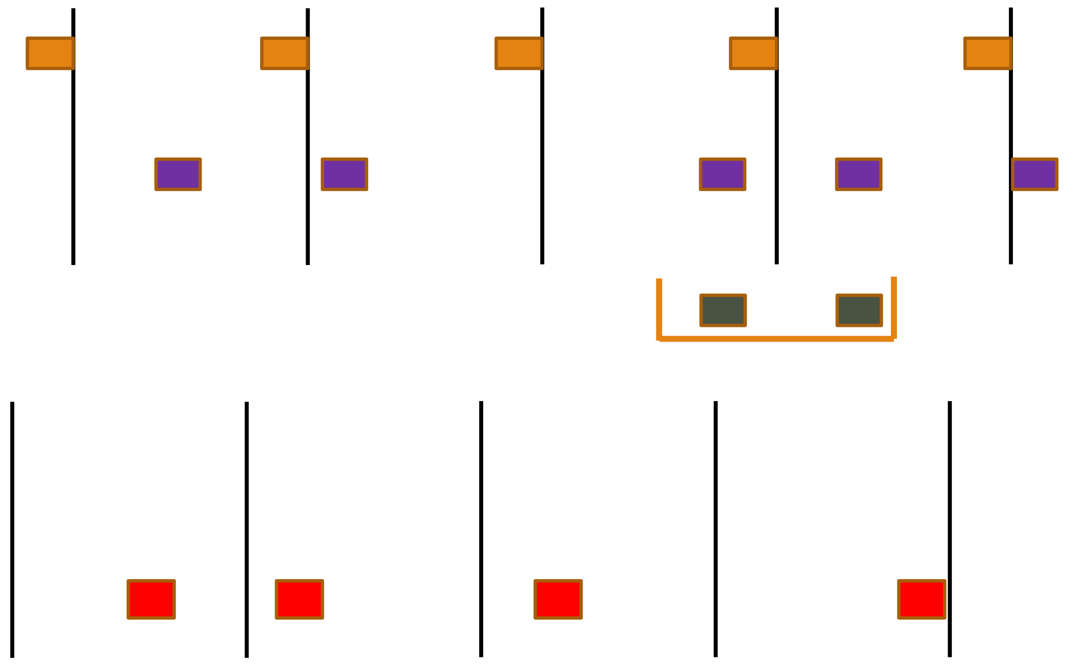
MSRP reservation = 2 frames per 1 ms interval

1 ms cycle time on IEEE Std 802.1Q-2018 Annex T CQF

CQF reservation = 4 frames per 1 ms cycle.

- In a typical case, **4x over-provisioning** is required to get from MSRP to Annex T CQF.

# CQF problems: Ingress conditioning


Generated stream: 1 frame per ms
As output with interference
RAP reservation = 1 frame per ms, burst = 2
1 ms cycle time on mCQF
CQF reservation = 1 frame per 1 ms.

- mCQF uses Paternoster byte counters, and more than 2 bins, so that one frame can be deposited in each bin. No overprovisioning, except to accommodate frequency differences.

# CQF problems: miscellaneous

- mCQF provides a means for adding a fixed delay along a path for every frame in a stream, in order to equalize delivery times along multiple paths.  This solves the problem of excess data at a multi-path merge point after the healing of a failed faster path.
- Having multiple cycle times allows faster streams to use smaller, faster-cycling buffers, instead of occupying lots of space in slower-cycling buffers.
- The paternoster counters allow strict syntonization to be relaxed, at a negligible cost in over-provisioning, given that Pulsed Queuing, by its nature, requires some over-provisioning as the cost of reduced calculation and configuration requirements.

# Pulsed Queuing Aggregation and Disaggregation

## PQ Aggregation and Disaggregation

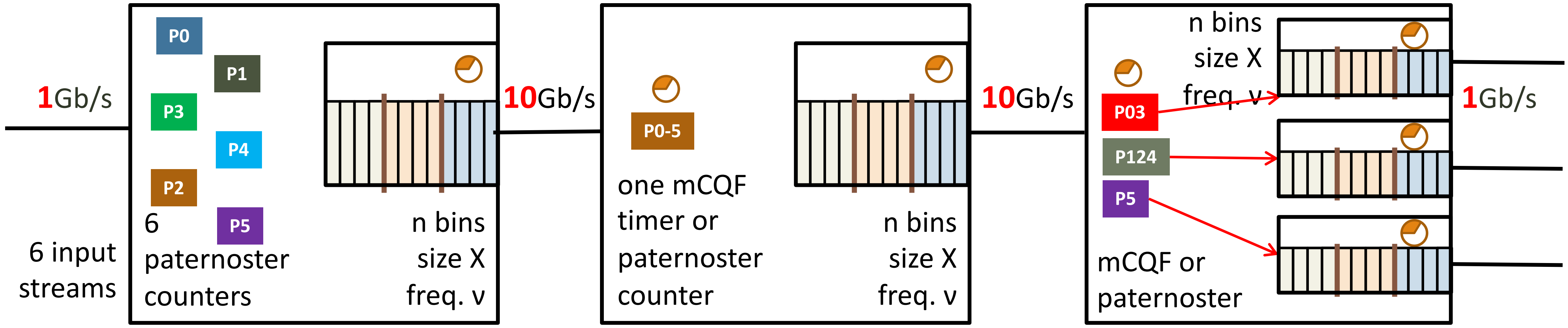
We will look at aggregation and disaggregation in three sections:

1. Aggregation without changing bin rotation frequency.
2. Aggregation with increased bin rotation frequency.
3. Performing both aggregation and disaggregation in a single node.

## 1. Aggregation with constant bin frequency

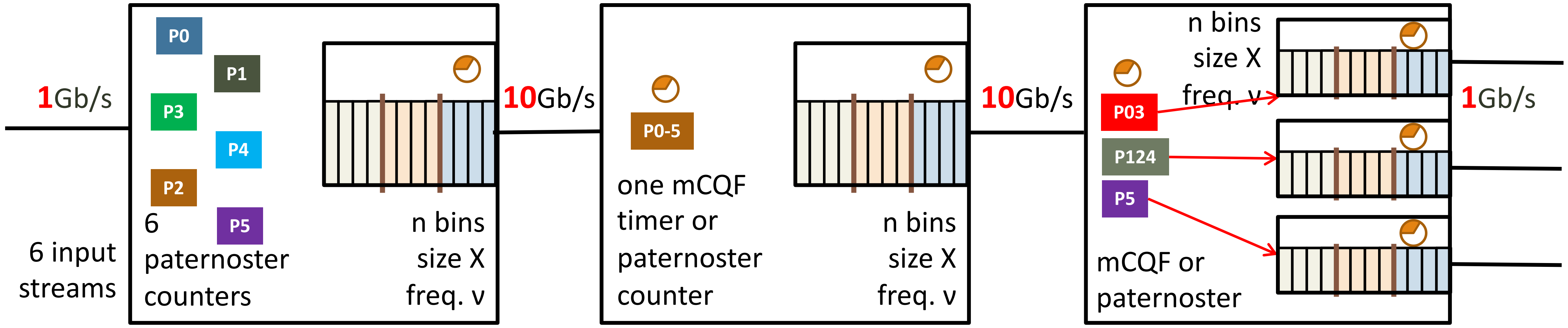
- We will not look at protocols for encapsulating streams.
  - Using .1CB Stream Identification, we can encapsulate them and use a single stream identification unit. (Encapsulation increases the aggregate bandwidth of the streams, and must be accounted for in the reservation.)
  - We can choose to not encapsulate, use one stream identification unit per component stream, and give all of the streams the same stream_handle.
- **As long as we do not shift any stream to a faster or slower bin rotation frequency, the only effect that aggregation has on mCQF or Paternoster is to change the amount of per-stream state**.
  - This works because aggregation/disaggregation does not alter the principle that frames that start in the same bin stay in the same bin.

# 1. Aggregation with constant bin frequency



- Streams can be split and merged arbitrarily.
- **Fewer counters needed**; it still works because all streams share one bin.
- Faster links mean that **more streams can be carried** in that one bin.

# 1. Aggregation with constant bin frequency

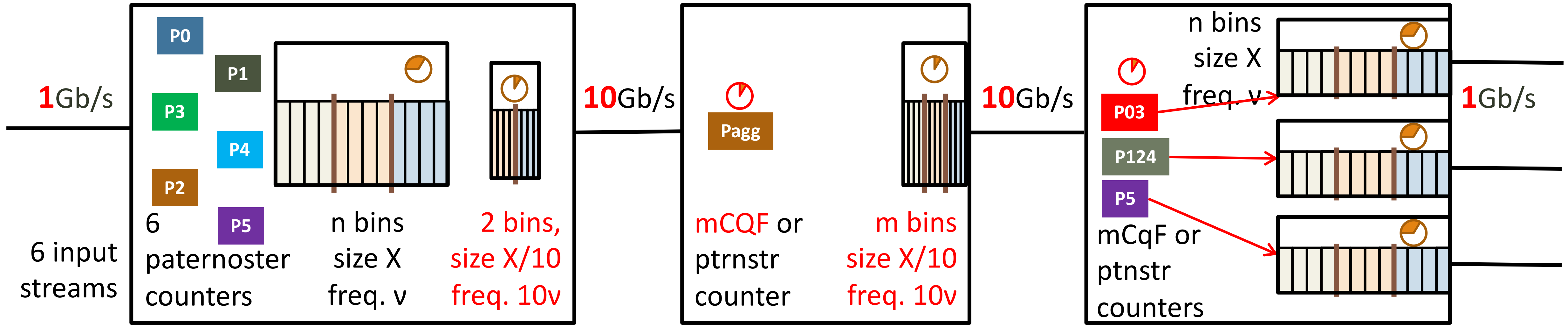


- Streams can be split and merged arbitrarily.
- **Fewer counters needed**; it still works because all streams share one bin.
- Faster links mean that **more streams can be carried** in that one bin.
- But, constant-frequency aggregation **does not improve latency**, and it **does not reduce buffer space**; it only reduces state.

## 2. Aggregation with increased bin frequency

- If we reduce the increase the bin rotation frequency (shorten the cycle time), then we can reduce the latency and buffer requirements in the intermediate nodes.

## 2. Aggregation with increased bin frequency

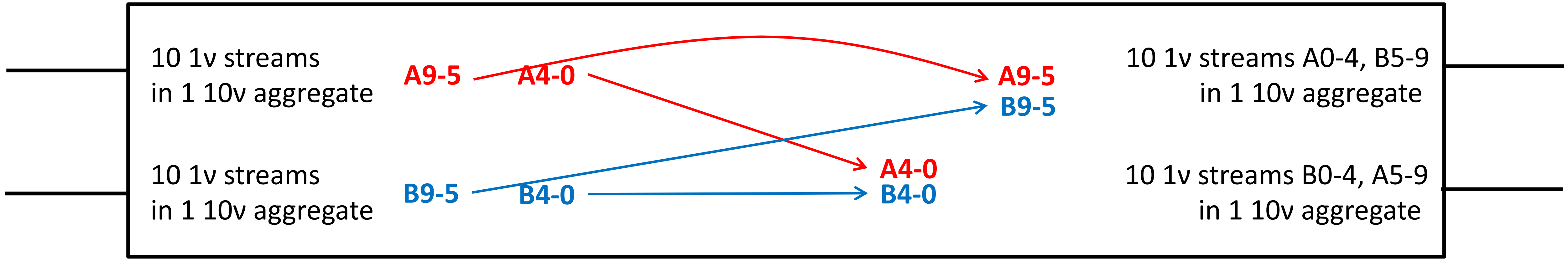


- Frames go from regular bins to tiny bins as they become eligible.
- Each small bin holds only one or a few frames of the aggregate stream.
- Any given component stream (P0, etc.) skips most of the small bins.
- **Buffer space and per-hop latency are reduced in intermediate node**.

## 2. Aggregation with increased bin frequency

- We are using a simple model for aggregating the streams; the original, slow bins are fed the smaller, faster bins in no particular order, except that we guarantee that each slow bin will be emptied on time.
- If the small bin/small cycles are kept in phase with the edge nodes, then mCQF timing can actually be used at the disaggregation point, dumping 10 fast bins into one slow bin.  Otherwise, Paternoster-like byte counters are required.
- Note that the order of the component streams' frames in the faster aggregate stream is arbitrary and variable; the only requirement is that they all meet their original large-bin timing.

## 3. Aggregation/disaggregation in one node



- Each component flow is spread, 1 frame per cycle; only one or two frames of the aggregate are present in the node at any one moment.
- The flows A0-A4 and B0-B4 arrive at the same time, but must exit spread out over 10 cycles. Same for A5-A9 and B5-B9, a little later.
- In this case, 5x buffers are required, and thus 5x latency. **We're back to our original problem with big buffers and large per-hop latency.**

## 3. Aggregation/disaggregation in one node

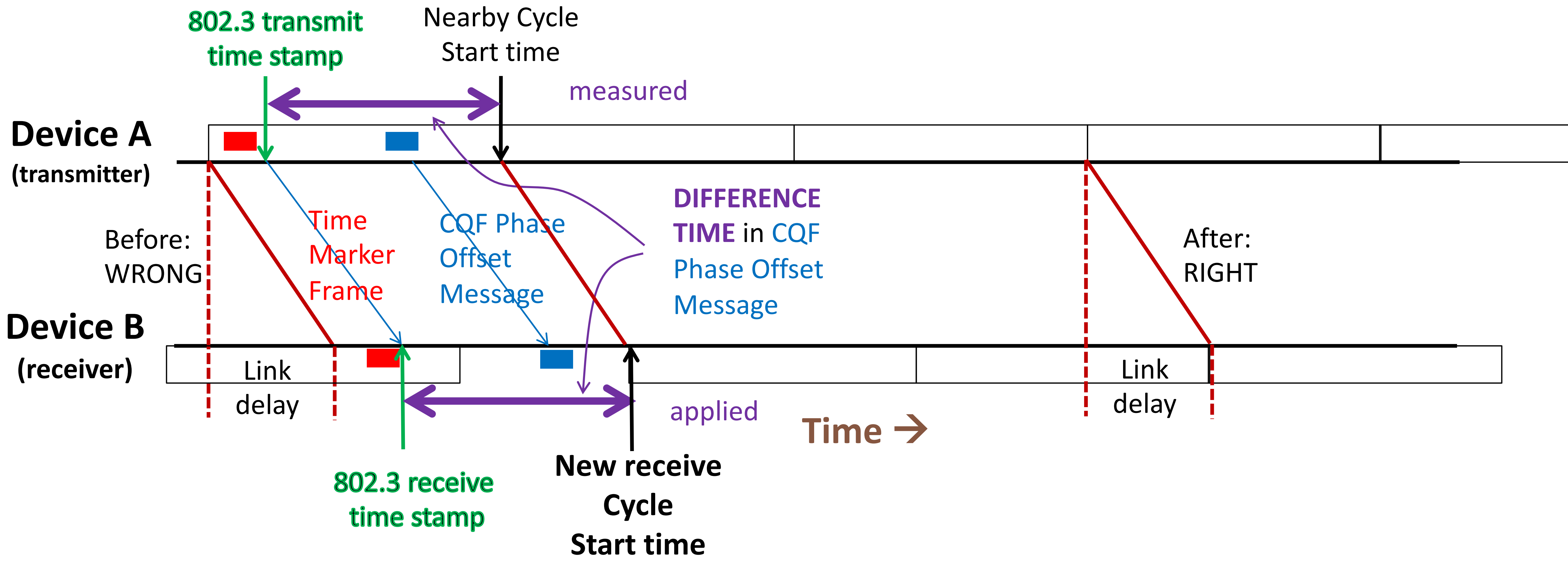
- **There Ain't No Such Thing As A Free Lunch.**
- Remember that the component streams' frames are in no particular order. All of the frames for a new aggregate can arrive at once, after a long pause.
- Therefore, when the component streams are split and merged, the buffers required are equivalent to those required if no aggregation had been performed.
- We can mitigate this, by paying close attention to fan-in, link speeds, and which streams aggregate in which ways. You can distribute the component flows' packets so that they better fit the requirements of the next re-aggregation.
- **This is called Asynchronous Traffic Shaping.**

# 3. Aggregation/disaggregation in one node

- Fundamentally, if you have a number of uncoordinated TSN flows passing through one port, you can be unlucky and have all of them arrive at once. In that case, all have to be buffered and they take a while to transmit. In the worst case, that happens at every hop. mCQF and Paternoster assume that worst case, and thus get away with trivial shaping (pulsed queues).
- If you take advantage of low fan-in, higher-speed links, and full knowledge of the characteristics of the flows, you can improve on the worst case. This is why ATS gives lower latency.
- **There Ain't No Such Thing As A Free Lunch.**

# CQF Transmitter-to-Receiver Synchronization

# The proposed solution



## The proposed solution: transmitter’s end

1. Device A transmits a Timing Marker Frame.  This can be any frame, but it must carry an identification value (TMFID) that changes with each transmission.
2. After transmitting the Timing Marker Frame, Device A recovers the time at which the first bit of the frame was transmitted from the hardware.  IEEE Std 802.3 Clause 90 specifies a method for accomplishing this.
3. Device A then transmits a CQF Phase Offset Message that contains a) the TMFID of a Timing Marker Frame, and b) the difference, in local time, between the start of a recent transmission cycle and the Timing Marker Frame transmission time recovered in step 2, above. (Specifically, [time of start of cycle] – [time of start of transmission].) Typically, this time difference would be expressed in increments of nanoseconds or finer.  The cycle chosen may be such that the time difference is positive or negative.

## The proposed solution: receiver's end

1. Device B receives a Timing Marker Frame.  It records the time of arrival of the first bit of the frame (again, IEEE Std 802.3 clause 90), and the TMFID of the frame.
2. Device B receives a CQF Phase Offset Message with a TMFID matching a recently-recorded Timing Marker Frame.
3. The time of reception of the Timing Marker Frame, plus the (signed) time difference carried in the CQF Phase Offset Message, is the local time at which the receive cycle that corresponds to the transmission cycle selected by the transmitter for reporting in the CQF Phase Offset Message, should have started (or should start, if in the future).  This establishes the cycle start time for the receiver's input gates in local time.

# Syntonization for Pulsed Queuing

## Syntonization for Pulsed Queuing

- mCQF, i.e. assigning received frames to PQ bins based purely on time of arrival, requires “perfect” syntonization (frequency lock) over the lifetime of a stream. We will discuss what this means a following slide.
- Paternoster, i.e. assigning received frames to PQ bins based on per-stream per-port byte counters, can be useful either with or without syntonization. If used without, then a degree of over-provisioning is required to prevent buffer overflow at the slowest port along the path.

## mCQF Syntonization

- In mCQF, frames travel in lock-step, always staying together in the same bin across the network. This requires that all nodes in the network be perfectly syntonized, in the sense that the difference between two nodes in the cumulative number of bin rotation cycles that have occurred must be bounded, no matter how long they have been running. Otherwise, bins can be delivered to a node faster than it can empty them, and TSN fails.
- The principles used in the IEEE Std 802.1Q-2018 scheduled transmission feature are adequate to ensure this, i.e. a synchronized start time and a rational frequency in seconds. Although synchronized time is not necessary for the operation of mCQF, synchronizing the clocks ensures that, in the long term, they are perfectly syntonized.
- See also new-finn-CQF-sync-method-09-21-v1.

## Summary

## Summary

- Pulsed Queuing, Multi-CQF and Paternoster are, for the most part, simply special cases of the more general Asynchronous Traffic Shaping.
- Syntonizing the nodes' clocks enables an alternative ATS implementation, Multi-CQF, that reduces the state machines required.
- Pulsed Queuing, Multi-CQF and Paternoster techniques clarify the tradeoffs available for ATS among compute time, configuration time, overprovisioning, and implementation cost.
- Given the fundamental limitations of the problem space, these tradeoffs can be useful.

## Working towards a PAR

## Elements to consider for a PAR

We should consider the following elements for a PAR scope:

1. Provide descriptions of CQF and Paternoster, but based on ATS, rather than 802.1Qbv scheduling.
2. Provide a protocol for synchronizing a CQF transmitter to its receiver.
3. Describe the operation of multiple classes of CQF service (multi-CQF).
4. Describe how to maintain TSN QoS when aggregating and disaggregating compound Streams.
5. Provide any managed objects needed to control these features.

# Thank you