Enhanced CQF and Stream Aggregation

Norman Finn
Huawei Technologies Co. Ltd
nfinn@nfinnconsulting.com
dv-finn-CQF-stream-aggregation-v01.pdf
July 11, 2022

Introduction

Purpose of this presentation

The purpose of these slides is to present some ideas related to Stream Aggregation. These thoughts are candidates for inclusion in the Annex to IEEE Std 802.1Q that is called out in the proposed PAR for IEEE P802.1Qdv Enhancements to Cyclic Queuing and Forwarding.

Enhanced Cyclic Queuing and Forwarding (review)

- Enhanced CQF is proposed for a PAR at this meeting (July, 2022, IEEE 802 plenary). Search "CQF" in the IEEE 802.1 public contributions pages for 2022 for a number of presentations on the subject.
- In brief, an IEEE 802.1Q class-of-service queue is (conceptually) divided into bins which are enabled for transmission sequentially, at a fixed frequency. CQF ensures that each bin will be emptied before the next bin is due for transmission. There are two ways of filling the bins:
  - **Time-based CQF** (multi-CQF) fills bins based on the time of arrival of each frame.
  - **Count-based CQF** (paternoster) fills bins based on per-Stream-per-output-queue byte counter state machines.
- Multiple queues on one output port can run at different frequencies, but they must be locked in particular phase relationships.

Stream Aggregation

- Stream Aggregation is **NOT** IEEE Std 802.1AX Link Aggregation.
- Stream Aggregation is the treatment, for at least the purpose of QoS, of all frames belonging to 1 or more TSN (Time-Sensitive Networking) Streams as if they belong to one single Stream.
  - Identification of the aggregate can be implicit, when multiple stream identification filters yield the same stream_identifier (see IEEE Std 802.1CB).
  - Identification can be explicit, when frames belonging to the aggregated streams are wrapped in outer header, matching a single identification filter.
- This presentation will not discuss any methods for explicit aggregation, nor discuss their effects on forwarding decisions, except to mention three of the many possible methods: IEEE Std 802.1ah MAC-in-MAC, IETF MPLS, IETF L2VPN.

Why is Stream Aggregation needed?

- In some applications, particularly internet service provision, but potentially for others, the number of Streams handled by a bridge can exceed practical limits on per-Stream resources such as Stream identifiers and QoS state machines.
- CQF, in particular, suffers as the number of Streams grows, as this increases the bin size, lowers the bin frequency, and increases end-to-end latency.
- For non-CQF TSN, for any given level of resource allocation efficiency, the time and compute cycles required to add or delete Streams increases much faster than linearly with the number of Streams.

How does Stream Aggregation help?

- Aggregating Streams reduces the number of Streams recognized, at least in the interior of a TSN network, thus reducing requirements for Stream identification, per-Stream QoS state machines, computation, and provisioning.
- Pre-allocation of an aggregated Stream allows its component Streams to be created or deleted with a minimum of computation and/or provisioning effort.
- Less obviously, Stream Aggregation can significantly decrease the per-hop delays of its component Streams. The result is to lower both the bridges' buffer requirements and the end-to-end delay.

Why is CQF important to aggregation?

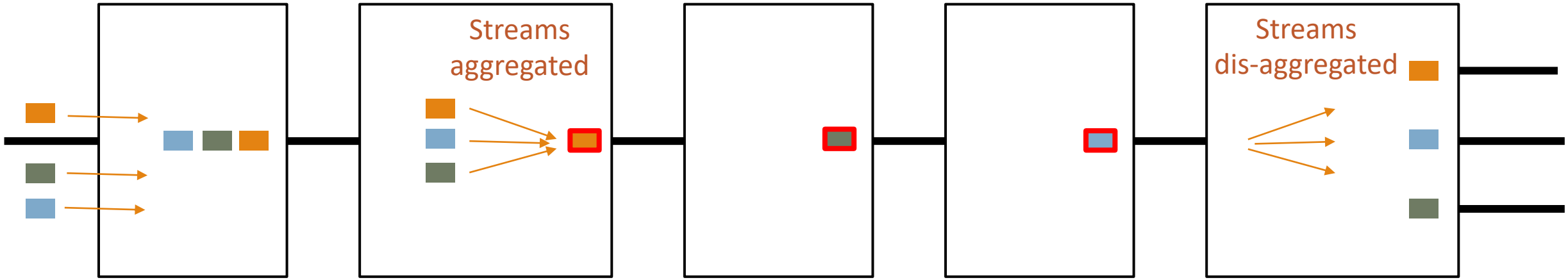
- If the Aggregated Streams use CQF, whether time-based or count-based, then the original burst/gap characteristics of the component Streams are maintained very well for the length of the aggregation. This minimizes the buffering required when the aggregation is dissolved.
- More to come in later slides, after we develop some concepts.

Per-hop delay

Decreasing per-hop delay

- We do **not** assume that TSN Streams are coordinated at their sources in order to avoid collisions at output ports in the network; such coordination is possible, but computationally difficult in the general case.
- This means that, for every Stream, there is typically one frame from that stream sitting in a buffer in every hop along its path. The end-to-end delay (not counting link and transmission delays) is then inversely proportional to the frame rate. (This relationship is obviously true by design for CQF, but it also applies to other queuing methods.)

Decreasing per-hop delay



3 Streams
3 stored frames/hop
1k frames/s/Stream
1ms delay/hop

3 Streams
3 stored frames/hop
1k frames/s/Stream
1ms delay/hop

1 Stream
1 stored frame/hop
3k frames/s/Stream
333µs delay/hop

1 Stream
1 stored frame/hop
3k frames/s/Stream
333µs delay/hop

3 Streams
3 stored frames/hop
1k frames/s/Stream
1ms delay/hop

Buffers required for basic operations

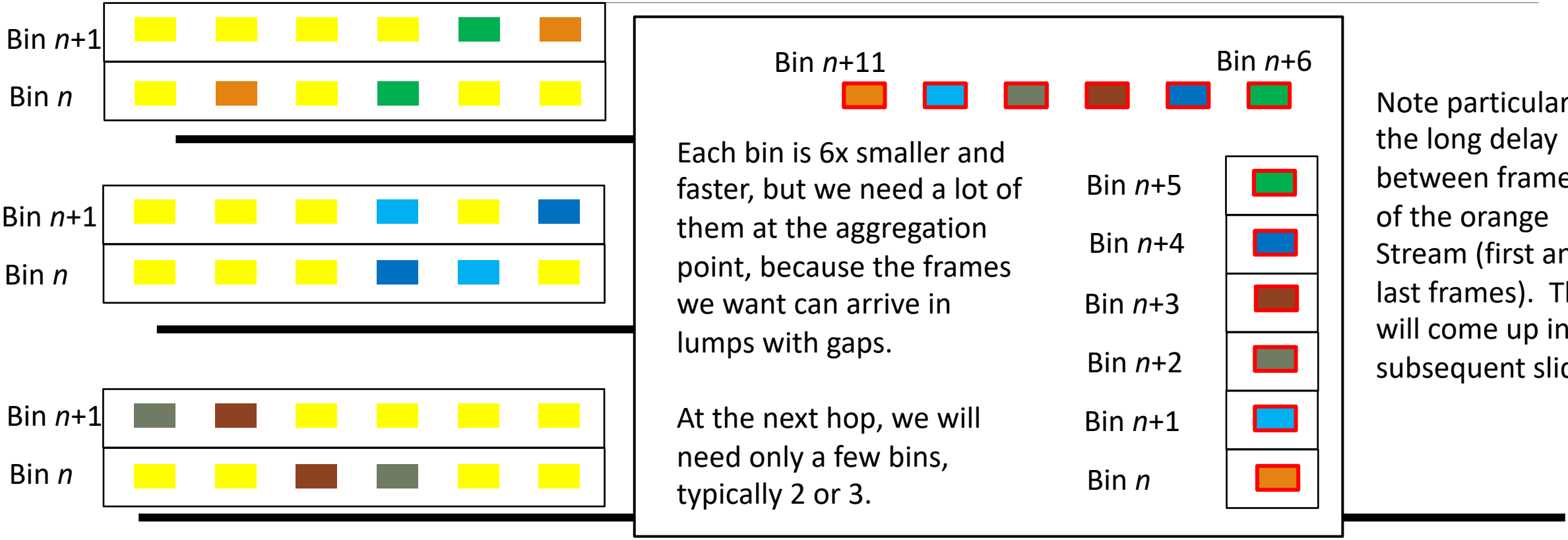
Buffering requirements

Extra buffering, and therefore extra delays, are encountered:

- Where Streams are merged to form an Aggregated Stream.
- Where an Aggregated Stream is dissolved and the Streams separated.
- Where an Aggregated Stream is reformed, with Streams both joining and exiting.

Aggregation merge buffers

6 Streams arriving from 3 inputs in 6ms CQF bins. This bridge aggregates them into one Aggregate Stream in 1ms bins. (Yellow frames are not part of this Aggregated Stream.)

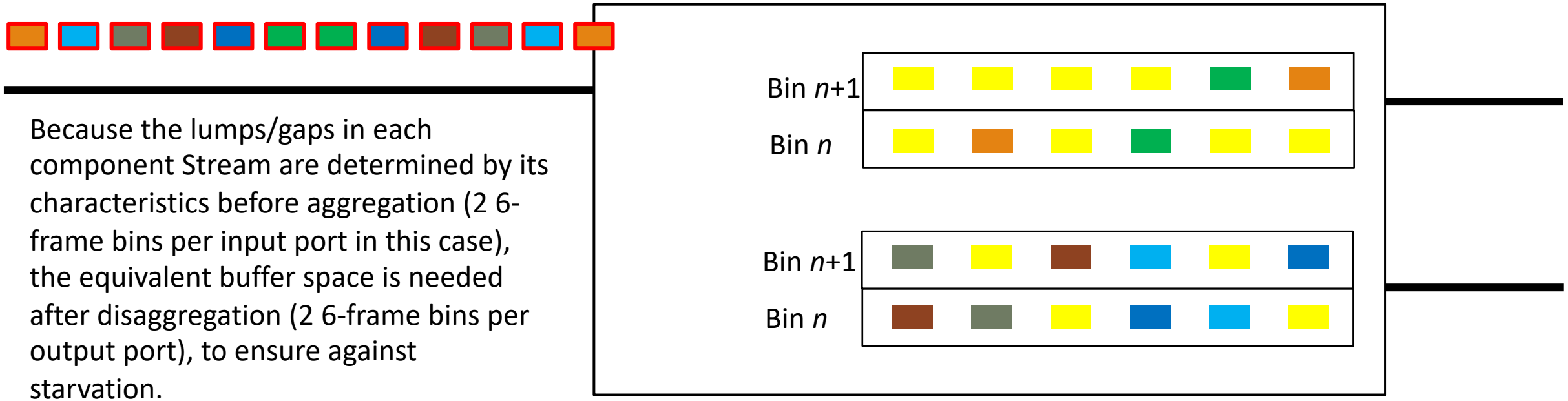


Note particularly the long delay between frames of the orange Stream (first and last frames). This will come up in subsequent slides.

It's the same issue with non-CQF; we need extra buffers when aggregating, to be sure that we'll never be starved for aggregated frames to transmit because of unfortunate coincidences in input timing.

Aggregation dissolution buffers

Since the Streams' original bin boundaries have been lost, the output bins are filled using count-based, not time-based, CQF.

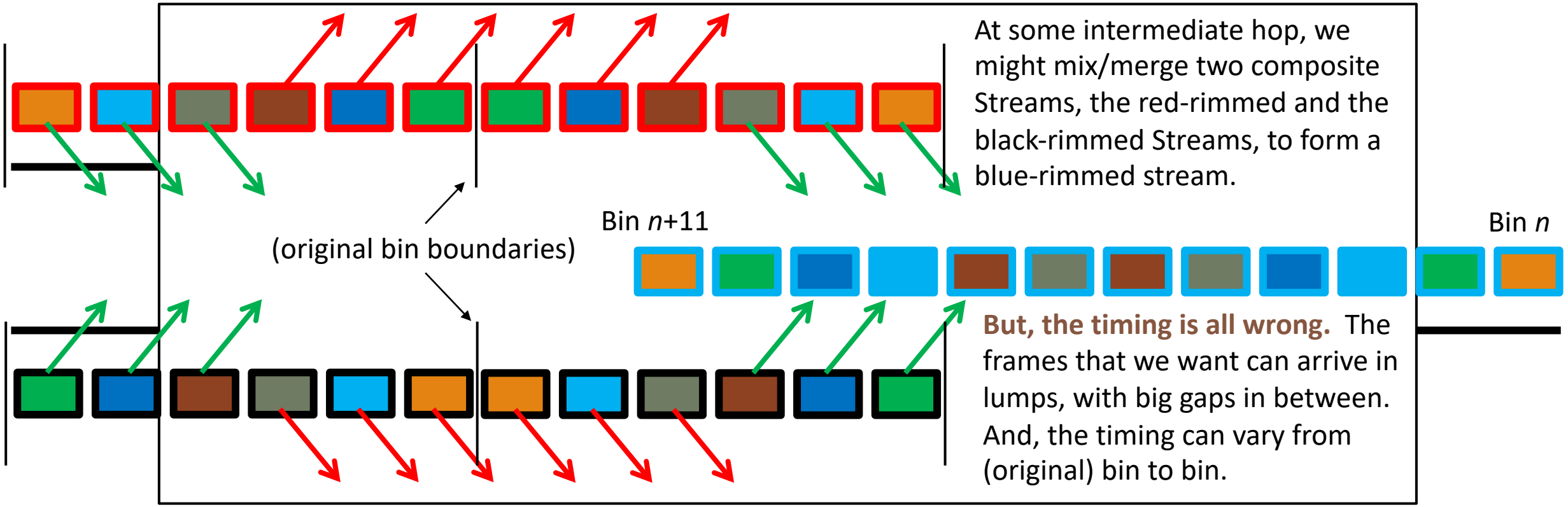


Because the lumps/gaps in each component Stream are determined by its characteristics before aggregation (2 6-frame bins per input port in this case), the equivalent buffer space is needed after disaggregation (2 6-frame bins per output port), to ensure against starvation.

Smaller or larger bins at different rates than the original Streams can be used, of course, as long as the allocated bandwidth is appropriate. But, the spacing of the frames for a given Stream was determined when the Stream Aggregation was formed, and that determines the buffer requirements, here.

Mix/merge buffers

Even though small bins are used for the Stream Aggregations, at a mix/merge point, the amount of buffer space needed is determined by the original characteristics of the Aggregated Streams, and in general, is equal to the buffer space needed by the Streams before aggregation.

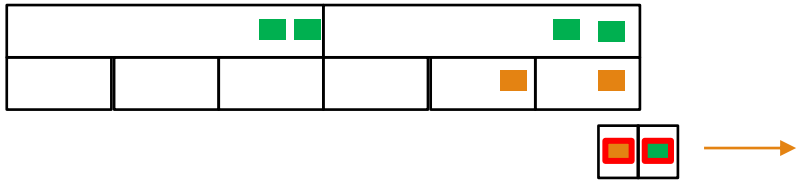


The sizes of the lumps/gaps, and thus the number of output bins required, are determined by the **original** Streams' characteristics (before aggregation). Count-based CQF fills the output bins.

Mix and dissolution buffers

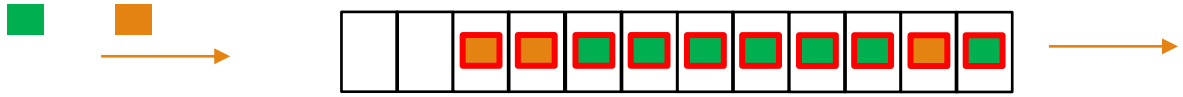
- It was mentioned that that amount of buffer space at a mix or dissolution point depends on the original characteristics of the component Streams.
- It also depends on how faithfully the Aggregated Stream's intermediate nodes maintain those characteristics.
- In time-based (syntonized) CQF, the Aggregated Stream's bin frequencies can be chosen to maintain those characteristics perfectly, and the required dissolution buffer requirements are the same size as for the original Streams.
- If count-based (paternoster) CQF is used, (bin assignments based on per-Stream byte counters), then the (relatively small) difference between the best- and worst-case delivery times from end to end of the Aggregated Stream will determine the extra dissolution buffer space required.

Faithfully maintaining Stream characteristics in the Aggregated Stream



- If the original Streams go into CQF bins as if they were not aggregated, and those bins are then drained into smaller, faster bins for the Aggregated Stream, always with aligned bin boundaries and integral frequency multiples (see multi-CQF writeups on IEEE 802.1 public site), then at least at the output from the merge point, the Aggregated Stream will faithfully reflect the original Streams' bin packing structure.
  - (Note that "drained into smaller, faster bins" can be accomplished without actually moving data or pointers, and without an extra small-bin delay, if one is clever.)

Faithfully maintaining Stream characteristics in the Aggregated Stream



- Alternatively, count-based CQF (paternoster) can be used to drop frames directly into the small buffers. The same amount of buffers space is required, as the input characteristics are still the same. This makes the merging job simpler, but the large bin boundaries are lost, so there is more variation within a single Stream, and more dissolution buffers are needed.
- With either form of CQF, the component Streams' frames do not make big jumps forwards or backwards through the small bins of the Aggregated Stream, so the extra dissolution buffers required are not excessive.

Faithfully maintaining Stream characteristics in the Aggregated Stream

- If any of the IEEE Std 802.1Q-2022 transmission selection algorithms other than CQF are used to carry the Aggregated Stream, then variations in one component Stream's activity can affect other Streams' delays along the path of the aggregation. This is because frames belonging to one component Stream can use other component Streams' unused bandwidth.
- This variation can be computed and extra dissolution buffer space allocated.
- If CQF carries the Aggregated Stream, this is either not possible (time-based CQF) or the variation is much smaller (count-based CQF).

Other points

Mix/merge madness

- If the order of the component Streams' frames were set in a particular order in each of the Aggregated Streams, then at the mix/merge point, the lumps/gaps could be reduced and many fewer extra buffers would be needed.
- This is similar, in principle, to the familiar (to some of us) proposal to allocate less than on frame per CQF bin to a Stream, and thus allow several Streams to share a single allocation in turn. When two such Streams fan in to one output port, there is a similar lump/gap problem.
- I have no plans to explore these possibilities, because if multicast Streams are included, or if the Aggregated Streams split and merge multiple times, it becomes computationally difficult to resolve the resultant conflicts in ordering requirements. It is also not trivial to initialize the transmitters, nor to adjust them to changing needs.
- Others are free to follow their own star, of course.

Swings and roundabouts

One may reasonably ask, “What have we gained with Stream Aggregation?” and, “At what cost?”

- If an Aggregated Stream is split and merged at every hop, absolutely nothing is gained.
- At the aggregation point, mix/merge point, and dissolution point, nothing is gained.
- In a network whose Streams have sufficiently unrelated paths, it may be than little can be gained.
- But, if many Streams have paths in common, for example, over a central interconnect ring, significant improvements can be made, both in TSN resources required, and in end-to-end delivery times.
- Explicit aggregation (adding a wrapper) is a common feature, but it can be costly. Implicit aggregation requires touching each bridge along the path of the Aggregated Stream. Perhaps some more creative solutions, based on IEEE P802.1CBdb, are possible.

Further work

- Mix/merge points affect the delay variation of component Streams, and thus affect the amount of dissolution buffering required. Further work is required to balance ease of implementation at mix/merge points against additional dissolution buffering at the network edge.
- It would be useful to look further at the case of merging Streams into compound aggregations. For example, do the requirements for mix or dissolution buffers change, depending on whether the composite Stream Aggregation is decomposed in steps, in reverse order, in some completely different order, or all at once?
- Further investigation is required on the question of what information is required for a mix or dissolution point to determine its buffer requirements.
- If Stream Aggregation becomes common, then it may become desirable to augment RAP (Resource Allocation Protocol).

A note on terminology

- I have used the terms “Stream Aggregation” and “Aggregated Streams” in this presentation.
- If the idea progresses in IEEE 802, then these terms will undoubtedly become overloaded. Possible uses of these and related terms include:
  - Aggregation of TSN Streams to conserve filtering/forwarding database resources.
  - Aggregation of TSN Streams to conserve QoS state machine resources. ← **primary use here**
  - Aggregation of non-TSN flows for either of those purposes.
  - Aggregations of Stream Aggregations, or component Aggregations in Aggregations.
  - Possible new forms of explicit (header wrapper) Stream Aggregation.
- Ideally, we will consider this overloading before selecting a term (or terms) for standard text that specifies any of these or similar uses.

Thank you