Multiple Cyclic Queuing and Forwarding

df-finn-multiple-CQF-0919-v02
Norman Finn
Huawei Technologies Co. Ltd
November 11, 2019

Abstract

Variants of Cyclic Queuing and Forwarding (CQF, IEEE Std 8021Q-2018 Annex T) are presented. CQF requires that all nodes run at the same frequency, but does not require that all output ports' cycles be in phase. Using three buffers (or more) allows for long links, and supports short/long path recombination. Running multiple instances of CQF on one port at different cycle rates can give good latency and bandwidth utilization for a mix of Streams with very different bandwidth requirements. These improvements require little or no alteration of IEEE 802.1Q, and like the original 2-buffer CQF, none require per-hop per-Stream dynamic state. These ideas are presented for possible inclusion into IEEE P802.1DF TSN Profile for Service Provider Networks.

1 Introduction

The remainder of section 1 defines the domain of interest of this paper. Section 2 provides a detailed timing model for Cyclic Queuing and Forwarding (CQF, IEEE Std 802.1Q-2018 Annex T). It shows how two, three, or more buffers can be used to manage the allocable bandwidth and per-hop delay, and why the systems' CQF cycles do not have to operate in phase. Section 3 shows how multiple instances of CQF can be run on the same output port, with different cycle times, in order to efficiently serve Streams with a wide range of bandwidth and latency requirements. Section **Error! Reference source not found.** lists a number of further augmentations that can be made to CQF to improve bandwidth utilization and worst-case latency.

This paper assumes the reader is reasonably familiar with CQF as described in IEEE Std 802.1Q Annex T. The frame timestamps used in this paper are described in IEEE Std 802.3-2018 clause 90. Preemption, or interspersed express traffic, is described in IEEE Std 802.3-2018 clause 99 and IEEE Std 802.1Q-2018 clause 6.7.2.

1.1 BL/ZCL Quality of Service

TSN (and IETF DetNet) supply a Quality of Service (QoS) to a critical data flow, or "Stream". This QoS is:

a. An absolute upper bound on the end-to-end latency to frames belonging to the Stream. (Bounded Latency, BL); and
b. A guarantee that no frames of the Stream will be discarded due to a buffer being full when the frame arrives at an intermediate hop (Zero Congestion Loss, ZCL).

This QoS, which we will call BL/ZCL, is made possible by a promise, made by the source of a Stream, to not exceed a contracted bandwidth and maximum frame size. This guarantee allows the network to run a resource reservation procedure that dedicates resources to a particular Stream (or sometimes, to a class of similar Streams) at every hop through the network, before the first frame of the Stream can be transmitted.

1.2 Continuous Streams

We can divide the Streams that can make use of the BL/ZCS QoS into two classes:

- Continuous Streams can be usefully characterized by a maximum frame size, and a maximum bandwidth.
- Scheduled Streams transmit on a regular, repeating schedule.

Note that these two categories do not encompass all possible data flows. Bursty, irregular flows are not Streams, in the sense that it is difficult, in the presence of multiple of these flows, to guarantee BL/ZCS except by gross overprovisioning and an extensive analysis of worst-case inter-Stream interference scenarios.

Scheduled Streams can be handled by using IEEE Std 802.1Qbv (now IEEE Std 802.1Q-2018 clause 8.6.8.4, Enhancements for Scheduled Traffic) to schedule frames in detail. They are of no interest to this paper. This paper Is concerned only with continuous Streams.

1.3 Store-and-forward

We will deal here only with store-and-forward systems, where whole transmission units are received, enqueued, and forwarded on another link. That is, we will not attempt to reconcile CQF with cut-through forwarding. We will, however, consider ways in which frames can be subdivided and/or preempted to reduce the size of transmission units.[1]

2 CQF timing model

We have two nodes, A and B. Both are running an instance of Cyclic Queuing and Forwarding on each of multiple ports, more-or-less as described in IEEE Std 802.1Q-2018 Annex T. We will

[1] The reader may also be familiar with the terms, “bulk streams” and “intermittent streams,” defined in 7.1.1 of IEEE Std 802.1CB-2017. These terms have meaning only in the context of choosing an algorithm for use by the 802.1CB sequence recovery function.

assume that nodes A and B are synchronized with each other in time to some accuracy that is significantly smaller than the CQF cycle time. We do not assume that the output buffers switch in synchrony; they can be out of phase.

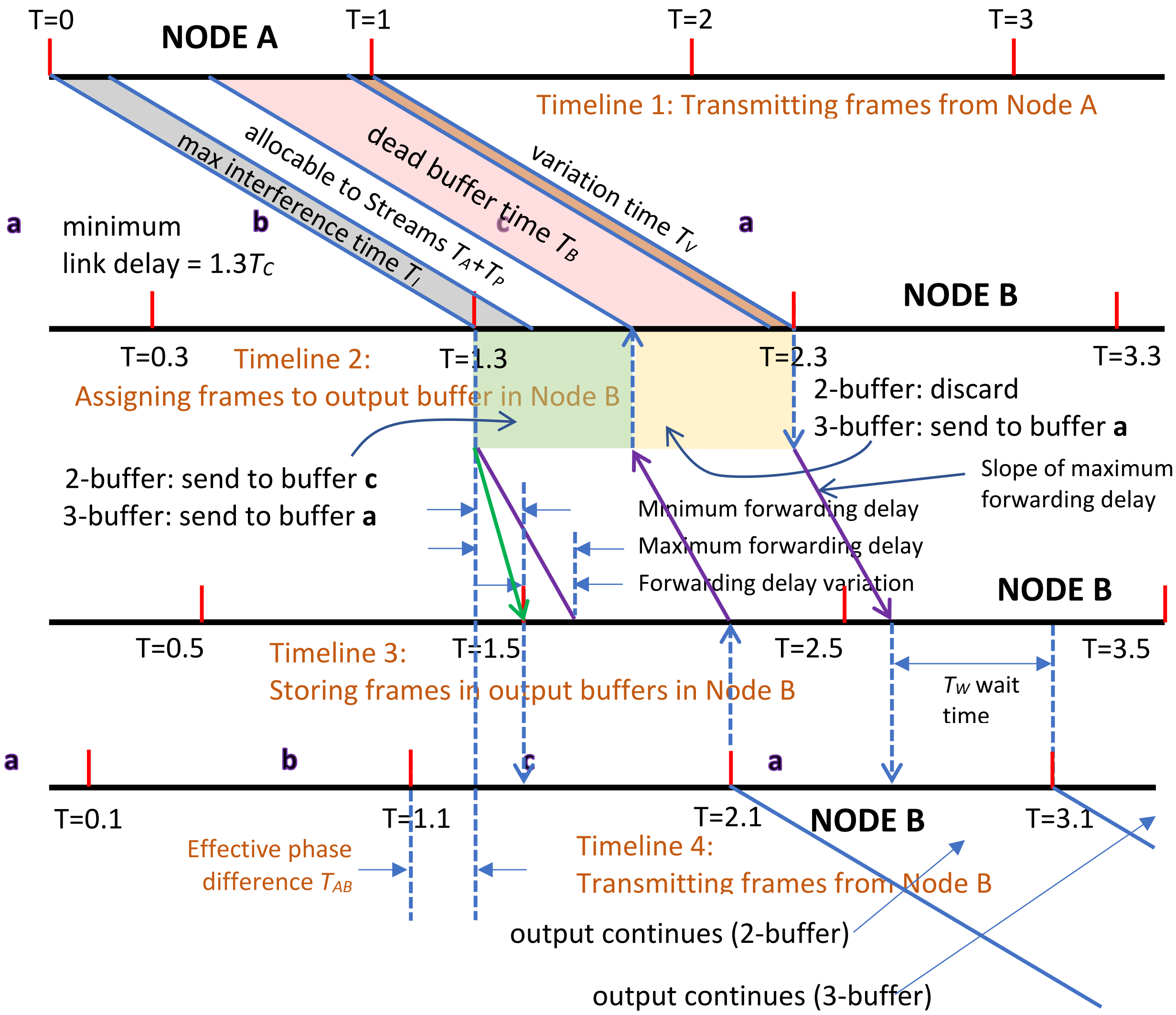
After a gate open/close event on a particular port, node A transmits all of the frames in one cyclic buffer towards receiving node B, not necessarily in a single burst. After some gap following the transmission of the last frame in the buffer, another gate open/close event is performed. At this point, it starts transmitting the frames from the next cyclic buffer. The gate open/close events in both nodes happen regularly, with the same period T_C. At the next hop, node B must be able to assign each received frame to a transmit buffer such that 1) frames that were in the same buffer in node A, and are transmitted on the same port from node B, are placed into the same buffer in node B; and 2) frames in different buffers in node A are placed in different buffers in node B.

Figure 1 shows an example of Cyclic Queuing and Forwarding. Node A and Node B are transmitting at the same frequency, but are offset by $0.1T_C$, as shown by timelines 1 and 4. In Figure 1, we use the following notation for time intervals:

T_C	nominal (intended) period of the buffer-swapping cycle
T_I	maximum interference from lower-priority queues, one frame or one fragment
T_V	sum of the variation in output delay, link delay, clock accuracy, and timestamp accuracy
T_A	the part of the cycle allocable to (reservable by) Streams
T_P	worst-case time taken by additional bytes if this traffic class is preemptable
T_B	end-of-cycle buffer dead time optionally imposed on node A by node B
T_W	wait time during which buffer is neither receiving nor transmitting frames
T_{AB}	effective phase difference between cycle start times for input from A and output from B

Following the definitions of output gates in IEEE Std 802.1Q-2018, the red ticks in timelines 1 and 4 in Figure 1 represent the earliest possible moment at which the first bit of the destination address of the first frame of the cycle can be transmitted. These ticks are driven by the synchronized clock. They are the basis for all cyclic buffer transmissions. If Enhancements for Scheduled Traffic (ETS, IEEE Std 802.1Q-2018 8.6.8.4) are used for controlling the output buffers, the ticks are the points in time when the output gate of one queue is closed, and the next queue's gate is opened. These are the points in time as programmed into the managed objects that control ETS. An implementation may need to schedule events in anticipation of the time specified in the managed objects in order to maximize throughput. Note that the preamble of an IEEE 802.3 Ethernet frame can be transmitted before gate open event.

Figure 1 *Reference timelines for time-based CQF*


T=0
NODE A
T=1
T=2
T=3
Timeline 1: Transmitting frames from Node A
max interference time T_I
allocable to Streams T_A+T_P
dead buffer time T_B
variation time T_V
a
minimum link delay = 1.3T_C
b
c
a
NODE B
T=0.3
Timeline 2:
T=1.3
T=2.3
T=3.3
Assigning frames to output buffer in Node B
2-buffer: discard
3-buffer: send to buffer **a**
2-buffer: send to buffer **c**
3-buffer: send to buffer **a**
Slope of maximum forwarding delay
Minimum forwarding delay
Maximum forwarding delay
Forwarding delay variation
NODE B
T=0.5
Timeline 3:
T=1.5
T=2.5
T=3.5
Storing frames in output buffers in Node B
T_W wait time
a
b
c
a
T=0.1
T=1.1
T=2.1
NODE B
T=3.1
Effective phase difference T_{AB}
Timeline 4:
Transmitting frames from Node B
output continues (2-buffer)
output continues (3-buffer)

2.1 Output timeline 1

Figure 1 shows an interference delay T_I (the gray area) between the gate event (the red ticks in Figure 1) and the transmission of the first bit of the first Stream frame's destination MAC address. The interference is from frames transmitted from lower-priority queues. It is equal to the time required for one maximum-length transmission unit over all lower-priority queues. That maximum transmission unit is either a maximum-length fragment, for preemptable lower-priority queues, or the maximum-length frame, for non-preemptable queues. The value of T_I depends upon the configuration of lower-priority queues.

It is possible that the class of service illustrated in Figure 1 is, itself, a preemptable class. In that case, a higher-priority class of service can preempt transmission of frames in this class.

Preempting a frame adds additional bytes to the resultant fragments, which must be accounted for when allocating bandwidth to a class of service. T_P represents the worst-case additional time required to transmit these extra bytes caused by preempting frames belonging to a CQF Stream. This value is always bounded. See below, section 3.2.

There can be some variation in the time from the selection of a frame for output in node A to the time stamp moment, when the first bit of the destination MAC address is transmitted (see IEEE Std 802.3-2018 clause 90). This is called output delay variation. The total time between the transmission of the first bit of the frame and the reception of that first bit at the next hop is called the link delay. Depending on the medium and the length of the link, there can be variations in link delay. The worst-case variation between the two node's clocks caused by accumulated frequency variations, asymmetrical links, etc., causes uncertainty between the transmitting and receiving nodes' clocks, and in the determination of the link delay. The inaccuracy in converting between IEEE 802.3 transmit and receive timestamps and the local clock that drives the gate open/close events also contributes to cycle accuracy. The worst-case combination of these four items, output delay variation, link delay variation, clock/frequency uncertainty, and timestamp conversion inaccuracies, is labeled, T_V.

All of the contributions to T_V are lumped together at the end of the cycle, even though contributions to T_V are made throughout the cycle.

As described below (section 2.4) the next hop can impose a buffer dead time T_B on this hop. This is a time at the end of the cycle, during which no frames can be transmitted from the cyclic buffer, so that the last frame of the cycle can be received earlier than the end of the cycle.

It is necessary, in order to know how much data can be transmitted in one cycle, that an implementation be able to transmit all of the frames in a cyclic output buffer together, at line rate, with no interference from lower-priority queues on the same output port. (Interference from higher-priority queues is described in section 3.2.) Given that is true, then the total time per cycle that can be used for transmitting Streams is:

$$T_A = T_C - T_I - T_P - T_B - T_V.$$

This T_A is a maximum, local to a particular output port on a node. It guarantees that the last frame of cycle (plus a possible preamble of the first frame of the next cycle) will be on the wire before the output gate closes. All of the components of T_A can be calculated by an implementation from its configuration and from knowledge of the implementation, except for T_B and parts of T_V. T_B is supplied by configuration, or by the node to which the output port is connected. (It is the maximum value over all receivers, if the output link goes to a shared medium). T_V can be supplied either by the time sync implementation, by configuration, by summing the contributions of node A and node B, or by the specification of a maximum allowed value by a standard or an equipment purchaser.

Note that T_A, as defined here, includes the entire transmission time of Stream data, including one 12-byte inter-frame gap and one 8-byte preamble for every frame. The preamble of the first frame of a cycle is counted in the previous cycle due to the way in which the output gates are defined in IEEE Std 802.1Q.

2.2 Receive timeline 2

The timeline at the receiving port is timeline 2 in Figure 1. The red ticks represent the earliest possible moment that the first bit of the destination MAC address of the first frame of a cycle can be received. In terms of IEEE 802.1Qci (IEEE Std 802.1Q-2018 clause 8.6.5.1 Per-Stream Filtering and Policing), a timed input gate must open no later than this point.

Received frames are assigned to a buffer based on the timestamp (IEEE Std 802.3-2018 clause 90) on the received frame.

A critical aspect of timeline 2 is its offset from timeline 4, the output timeline. This offset is shown as T_{AB} in Figure 1. It is clear from the figure that T_{AB} must be known in order to compute T_B and T_W. T_{AB} can be computed by 1) synchronizing the clocks of nodes A and B, and 2) measuring the link delay from node A to node B using PTP. Other methods are also possible.

Once T_{AB} is known, all of the timing relationships shown in Figure 1 can be computed. The phasing of the nodes' output buffer cycles certainly does affect the end-to-end latency of any stream, so that phasing must be known when the latency is computed. The end-to-end latency is no longer an integer multiple of the cycle time. It is even possible to adjust the phasing to favor certain paths through the network.

For a node B that is connected to and receiving cyclic frames from *n* other nodes, we have *n* assignment problems to solve, one for each input port on node B.

If a frame is received that straddles a cycle (first be in one cycle on timeline 2 of Figure 1, and end frame plus inter-frame gap plus a preamble time occurs in the next cycle), then either 1) some part of that frame was transmitted from node A outside the cycle window T_C, or 2) one or more of the constants, measurements, or calculations above is incorrect. Either way, the frame must be discarded, or else it can cause disruption of delivery guarantees farther along in the network.

2.3 Storing frames timeline 3

The timeline at the point where frames are stored into an output buffer is timeline 3 in Figure 1. The red ticks on timeline 3 mark the earliest point at which the first frame transmitted from a particular buffer could reach the output buffers (neglecting transmission time on the input medium). These ticks are offset from timeline 2 by the minimum forwarding delay, required to forward the frame from the input port to the output queue. The maximum forwarding delay is also shown. The forwarding delays shown in Figure 1 include the time to install the frame in the output buffer and for its presence to filter through to the point that it can be selected for output.

There are two possible buffer assignment methods shown in Figure 1: the two-buffer method, in which the frames received from node A buffer a are assigned to buffer c in node B, and the three-buffer method, where those same frames are assigned to buffer a in node B. The slope of the maximum forwarding delay allows us to compute the latest moment at which frames received from buffer **a** on node **A** can be stored into buffer **c** on node **B**. The shaded areas just below timeline 2 in Figure 1 show the time windows for buffer assignment. If two output buffers are used, then frames received from buffer **a** on node **A** can be assigned on input (timeline 2) to buffer **c** only as long as they are assured of being placed into buffer **c** before node **B** starts transmitting buffer **c**. As shown, frames from buffer **a** can be assigned to buffer **a** (three-buffer mode) during the entire length of the cycle on timeline 2. Time T_W in Figure 1 is the time during which, in three-buffer mode, buffer **c** is holding frames, neither filling nor emptying. In 3-buffer mode, the dead buffer time T_B is 0, and T_B, the allocable transmission time, encompasses both the T_B (white) and T_B (red) regions in Figure 1.

2.4 Calculation of T_B

Timeline 3 in Figure 1 shows the calculation of T_B, which applies only to two-buffer mode. The starting point of is the moment that the output cycle starts (the tick on timeline 4), backed up by the worst-case forwarding delay. This is the last moment on timeline 3 that a frame can be assigned to buffer **c** in the example in Figure 1. The end of T_B is the end of the cycle T_C, less the variation time T_V. In three-buffer mode, T_B is zero.

T_B is only known to node B. Its effect on the allocable bandwidth T_A must be taken into account when admitting new Streams. If a network uses a peer-to-peer control structure using, e.g. IEEE Std 802.1Q-2018 MSRP, then the value of T_B must be made available to the previous node A so that node A does not exceed the reduced T_A.

There are many ways to deal with this issue. Here are three:

1. The value of T_B can be propagated backwards to the previous node, either via management or via an extension of the reservation protocol.

2. A node can compute the value of T_B and decide whether to employ 2-buffer or 3-buffer mode, depending on how much bandwidth has been allocated, so far. This, of course, can change previously-computed Stream's end-to-end latency.

3. All nodes in a network can be configured with a reasonable maximum value for T_B. If a particular input/output port pair on a particular node computes a value for T_B that exceeds this maximum, then 3-buffer operation is required.

2.5 Transmitting frames timeline 4

Depending on whether two-buffer or three-buffer mode is used, one can trade off reduced total available bandwidth against per-hop delay. Timeline 4 in Figure 1 shows the two options for the choice of which output cycle in node B is used to transmit frames that were transmitted from buffer **a** in node A.

2.6 More than 3 output buffers

The discussion over Figure 1, so far, assumes that the variation in forwarding delay is small, relative to T_C. If this is not the case, node B can use more than 3 output buffers, and assign received frames to buffers whose output is scheduled far enough ahead in time to ensure that, in the worst case, they will arrive in the buffer before the buffer begins transmitting. This works only because the buffer assignment decision is made based on time-of-arrival of the frame at the input port, not the time-of-arrival of the frame at the output port.

In certain situations, e.g. when Stream is split and traverses two paths of different lengths using IEEE Std 802.1CB Frame Replication and Elimination for Reliability (FRER), it can be desirable to purposely delay a Stream's frames in order to match the total delay for the Stream along the two paths. In this case, more than 3 output buffers can be allocated, and used to impose a delay of an arbitrary number of cycle times T_C on every frame.

This author claims, without a demonstration, that it is not difficult to implement CQF so that each output port in a node, and each output port along the path of a Stream, can have a different number of buffers, whether 2, 3, or 50. Not only that, but one flow can use 3 buffers on an output port, while another flow, which needs a path-matching delay, can use 12 buffers on the same port. (Of course, this would require per-flow configuration.)

3 Multiple CQF classes of service

3.1 Multiple T_C model

With CQF as it is described in IEEE Std 802.1Q-2018 Annex T, we are limited to a single class of service (a single value of T_C) and to 2-buffer operation, only. We have already discussed 3-buffer (or more) operation. We will now discuss the simultaneous use of more than one value of T_C on the same output port.

It can be difficult to pick a single value of T_C for a network. If the chosen value is small, then only a few Streams can be accommodated on any one port; all frames for all Streams must fit into a single T_C period. If the value chosen for T_C is large, then more Streams can be accommodated, with a wide variation in allocated bandwidth, but the larger T_C increases the per-hop latency. In the ideal case, of course, every Stream would have a T_C value chosen so that exactly one frame of a Stream is transmitted on each cycle T_C.

While this is not always possible, we can apply multiple values of T_C to a single output port, as shown in Figure 2.

Figure 2 Multiple T_C *values*

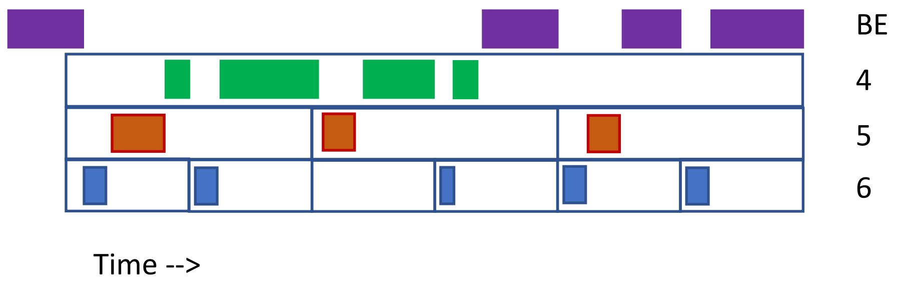
h																								3
f								g								f								4
d				e				d				e				d				e				5
a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	a	b	c	6

In Figure 2, we have a schematic timeline. We are running four values of T_C simultaneously. The fastest (call it, "T_{C6}") runs at the highest priority (6). T_{C5} is slower by a factor of 4 from T_{C6} in this example, and its buffers run at priority 5 (less important than priority 6). T_{C4} is slower by a factor of 2 from T_{C5}, and by a factor of 8 from T_{C6}. T_{C3} is 24 times slower than T_{C6}. The letters in Figure 2 label which buffer is output during the cycle. There are 9 buffers a through i (buffer i is not shown). In this example, priority 6 uses three buffers, because the timing is tight; the others use two each.

We assume here that the receiver of a frame can identify the particular CQF instance (T_C value) to which the frame belongs by inspecting the frame. A TSN bridge could use the L2 priority of the field, for example. An IP router could use the DSCP. IEEE Std 802.1CB and IEEE Std 802.1Q provide for the use of other fields in the frame, e.g. IP 5-tuple.

Since the total bandwidth of the link is not oversubscribed by Streams, each cycle, fast high-priority and slow low-priority, is guaranteed to be able to transmit all of its frames within the duration of its cycle. For example: If 50% of T_{C5} is reserved, and 30% of T_{C3} is reserved, then 80% of the total bandwidth has been reserved, leaving only 20% for other Streams, best effort traffic, and dead time. This is shown in Figure 3, where we illustrate the timing of transmission of frames from three levels of CQF and the best-effort (BE) level. Note that CQF traffic can be delayed within its window by interference from both higher priorities (e.g. the first priority 4 frame) and lower priorities (e.g. the first priority 6 frame), but that it will always get out before the window closes, assuming that the bandwidth is not oversubscribed.

Figure 3 Transmission timing



As is normal with CQF, a given Stream is allocated a fixed number of bits that it can transmit per cycle T_{Cn}. Each Stream is assigned to the highest-numbered (fastest) CQF instance such that, at the Stream's bandwidth and frame size, the Stream is fast enough to occupy space in every buffer at that level. Then, CQF will maintain one or two frames in its buffers per Stream, the best possible latency is given that Stream, and the buffer space is not wasted in unused cycles.

Of course, it is the "best possible" latency only to a certain extent. The potential mismatch between the Stream's frame rate and frame size to the available values of T_{Cn} requires some overprovisioning.

Streams are allocated to, and thus use up the bandwidth available to, each cycle separately. Any cycle can allocate up to 100% of the bandwidth of that cycle's T_A, but the percentages allocated to all of the cycles must, of course, add up to less than 100%. The total amount of buffer space required depends on the allocation of Streams to priority values. If all Streams are slow and are allocated to T_{C4} up to a total of 100%, then full-sized buffers must be used for buffers h and i. If all Streams are fast and are allocated to T_{C6}, then only three small buffers are used—buffers a, b, and c are rapidly re-used.

NOTE—There are many ways to allocate buffer space to individual frames. Running CQF at 5 levels does not increase the buffer memory requirements beyond that of 1-level CQF. Allocating bandwidth to slow cycle times uses more buffer space, of course, because frames dwell for a longer time. This is inherent in any scheme that offers comparable low-bandwidth high-delay service.

Given the ideal allocation described, each Stream is allocated one frame in each cycle of one row. It thus gets the optimal latency for its allocated bandwidth, which may be somewhat oversubscribed. If the end-to-end latency requirements of the Streams permit, a Stream can be assigned to a slower (lower-numbered) cycle. This will reduce the overprovision factor, since the overprovision factor depends on the number of frames per cycle. It also increases buffer usage, of course.

Any such overprovision can equally be thought of as an increased latency for that same Stream. That is, if that oversubscribed Stream was the only Stream, then the T_C cycle time could be shortened to exactly the point of 1 frame per cycle, with 0 overprovision, and thus give a faster latency. Overprovision = lower latency, in this case.

The maximum reserved bandwidth is supported by allocating a Stream multiple frames per cycle, as allowed by the Stream's required end-to-end latency, thus minimizing overprovision.

3.2 Preemption and interference

Frame preemption is described in IEEE Std 802.3-2018 clause 99 and IEEE Std 802.1Q-2018 clause 6.7.2. Not all of the bandwidth in a cycle T_C can be allocated. The smaller the cycle time, the greater the impact of the interference time (T_I in section 2 and Figure 1) on the allocable bandwidth.

T_I is equal to the worst-case transmit time for a single transmission from a lower-priority queue. This interference can occur only at the beginning of a cycle. Since this value must obviously be bound, it places a requirement, that must be enforced, on all lower-priority queues that they either have a maximum frame size or that frame preemption is applied to the lower-priority queues. If preemption is used, the maximum interference is the maximum fragment size (about 150 bytes, see IEEE 802.3). The interference time is shown as a gray parallelogram attached to timeline 1 in Figure 1.

The other time is the preemption time T_P, which applies only to Streams that are preemptable. This case is not typical, but is possible if a large fraction of the available bandwidth is to be assigned to one or a few high-bandwidth Streams, and lower-priority Streams use larger frames. T_P is the product of (the maximum number of highest-priority transmission windows that can open during a single window for the level being computed) * (the per-preemption penalty). Thus, in Figure 2, if priority 4 is preemptable, then there are 8 level 6 windows that can open. This means that there can be 8 preemption events during one level 4 window, so the total preemption time T_P is 8 times the preemption penalty. (It doesn't matter which specific frames are preempted; only how many such events occur.) The preemption penalty is the number of bytes added when a frame is preempted, which is 4 (CRC on preempted fragment) + 20 (inter-frame gap) + 8 (preamble for continuation fragment) = 32 bytes.

3.3 T_C computation

If the time per cycle that is allocable to Streams is T_A, then we can now state the computation for T_C, given T_A, or for T_A, given T_A, at each level in Figure 2:

$$T_C = T_A + T_P + T_I + T_B + T_V$$

The sum of all Stream's bits-per-cycle allocation must be less than or equal to T_A.

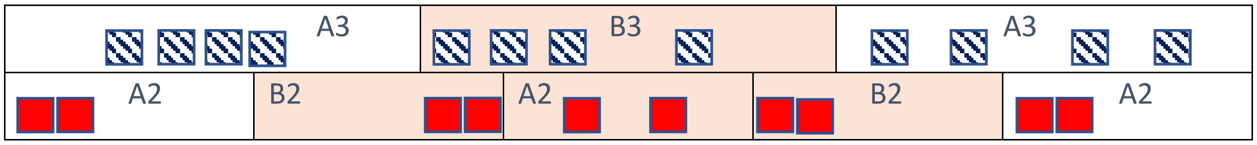
IEEE Std 802.1Q-2018 Annex T, assumes the 2-buffer scheme, and so assumes that T_B and T_V are small enough and T_C large enough to leave a useful T_A. Assuming that one's goal is the smallest possible T_C:

a. T_B can be eliminated by using the 3-buffer scheme.
b. Implementation steps can be taken to reduce T_V. This may include steps to reduce the variability of the forwarding delay, the delay between selection-for-output and first-bit-on-the-wire at the previous hop, or increased accuracy of the synchronized clock.
c. T_I can be reduced by restricting the maximum frame size of lower-priority Streams, or by enabling frame preemption.

3.4 Why integer multiples for T_C?

The ideal would for each Stream S to have its own T_{CS} that gives no overprovision. But, that winds up being equivalent to a per-Stream-shaper solution such as Asynchronous Traffic Shaping or IntServ. The reason can be seen in Figure 4.

Figure 4 Variable $\mathrm{T_C}$



In Figure 4, we have allocated 40% of the link bandwidth to the solid red Stream in cycle 2, and 50% of the link bandwidth to the blue striped Stream in cycle 3. The cycles do not line up with an integral number of faster cycles in each period of slower cycle. Since we cannot predict exactly where, during a cycle, frames can be emitted, we can get the situation shown, in the shaded boxes. Buffers B2, A2, and then again, B2 emit their frames (at high priority) at the indicated times. Even though the solid red Stream takes up only 40% of each level-2 cycle, it can output 6 frames over the course of cycle B3, thus taking up 60% of the bandwidth during that period. There is, therefore, 110% of the bandwidth that must be output during the period that B3 is transmitting. B3 cannot output all of its data. Some of it must be somehow delayed, but there is no place to put that data. TSN fails.

Having an integral number of cycles at each layer fitting into one cycle at the next-slower layer ensures that the lower-priority, slower cycle, will always have sufficient time to output all of its frame, because the problem in Figure 4 is avoided. It also bounds the number of preemption events that can steal bandwidth from a given priority level.

4 Other issues

4.1 Frame size problem

The above discussion has largely assumed that each Stream consists of frames of a uniform size, equal to the Stream's maximum frame size. Of course, this is not always true.

The advantage of uniform frame size is that, in the ideal case, one can allocate a Stream one frame per cycle, and choose the cycle time and/or the Stream's bandwidth reservation so that there is no wasted bandwidth. Similarly, if we imagine that a Stream alternates frames of 4000 bit times and 800 bit times, we can allocate 4800 bit times per T_C and still get perfect results.

But, in a service provider situation where we are allocating a certain bandwidth per customer, but the frame sizes are essentially random, things are not so simple. Let us suppose that the maximum frame for a Stream is 13000 bit times, which is approximately equal to a maximum-length Ethernet frame, and that the cycle time T_C = 100µs. 13000/100µs = 130 Mbits/s. But, allocating a bandwidth of 13000 bits/T_C will not give the Stream 130 Mb/s. In the worst case, one 13000 bit frame followed by one minimum-length frame = 672 bits, the Stream gets (13000+672)/(200 µs) = 68.36 Mb/s.

We could overprovision the Stream by a factor of almost 2, keep the same T_C, and get minimal latency. However, we could also assign the Stream to a longer T_C. In the worst case, there are (13000–8) wasted bits in each cycle. Therefore, we can guarantee 130 Mb/s using a cycle time of 500µs by provisioning (5*13000 + 13000 – 8)/(5*100µs), or 156 Mb/s, which is a 20% overprovisioning, rather than a 90% overprovisioning, at the cost of five times the per-hop latency.

This overprovisioning/latency tradeoff is only needed for Streams that have variable frame sizes, such as service provider Streams. But, for those Streams, the lengths of the links may be a larger source of latency than the queuing delays, so the situation may not be so bad. Also, any unused bandwidth is available to non-TSN data, so overprovisioning may not be a serious concern.

4.2 Bundling

IEEE Std 802.11n combines a number of Ethernet frames into a single transmission unit, in order to minimize the number of times per second a different transmitter starts sending data. Similarly, each CQF Stream, on ingress to the TSN network, can be run through a "sausage maker". That is, frames can be encapsulated using a scheme that combines and/or splits frames into uniform-sized chunks (sausages), either small or large, that can be carried end-to-end through the TSN network, then split out into their original form. This means that overprovisioning due to the mix of frame sizes is reduced to that required by the encapsulation, itself. (In fact, that overhead can be negative, if small frames are bundled into large transmission units.)

4.3 Tailored bandwidth offerings

We can note that, in a service provider environment, overprovisioning can be almost eliminated by a combination of 1) bundling (4.2) and 2) offering the customer only a specific set of choices for a bandwidth contract, corresponding to the values of T_C implemented in the provider's network.

4.4 Overprovisioning is not always bad

Overprovisioning the bandwidth (allocating more of T_A than is necessary) is not always a bad thing:

a. Allocating a Stream to a higher priority (smaller T_C) than it needs reduces its worst-case latency. This may be necessary to meet a Stream's end-to-end latency requirement. That is, one can overprovision the frame rate in order to obtain a reduced latency. The unused bandwidth is still available for best-effort traffic. Not all TSN transmission selection schemes have this feature.
b. If the total bandwidth required by critical Streams is relatively low, using faster-than-necessary T_C values will both improve latency and reduce buffer requirements in the network. The allocated-but-unused bandwidth is still available to best-effort traffic, and thus may be of no consequence.

4.5 Relationship to current standards

Multiple instances of CQF, or for that matter, using more than a very few buffers per instance, will quickly use up the 8 buffers that IEEE Std 802.1Q-2018 provides timed output gates for. Some effort would, therefore, be required to reconcile multiple CQF with IEEE Std 802.1Q. If the idea proves useful, however, this author does not believe that they would be difficult to reconcile.

4.6 Fundamental CQF pros and cons

The obvious downside of CQF is that it requires clock synchronization and per-port time-based gating. On the other hand, CQF requires no per-Stream per-hop active state machines. A new stream can be provisioned by a network controller without any interaction between the network controller and any of the network's relay systems, except for configuring one system for ingress policing. Furthermore, calculation of the worst-case end-to-end latency is trivial, and the calculation made for one allocated stream is never affected by any other allocations or deallocations.