C802.1Qdv/D0.1
January 19, 2023

(Amendment to IEEE Std 802.1Q™–2022 as amended by
IEEE Std 802.1Qcz™–2022)

Proposed text for Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment: Enhancements to Cyclic Queuing and Forwarding

Unapproved contribution, prepared by
Norman Finn, Finn Consulting, Inc.

Member of the
LAN/MAN Standards Committee
of the
IEEE Computer Society

.

Individual contribution to IEEE 802.1.

Abstract: This amendment enhances Cyclic Queuing and Forwarding. It specifies a transmission selection procedure that organizes frames in a traffic class output queue into logical bins that are output in strict rotation at a constant frequency; a procedure for storing received frames into bins based on the time of reception of the frame; a procedure for storing received frames into bins based on per-flow octet counters; and protocol for determining the phase relationship between a transmitter's and a receiver's bin boundaries.

Keywords: CQF, Cyclic Queuing and Forwarding, IEEE 802.1Q™, LAN, local area network, Time-Sensitive Networking, TSN, Virtual Bridged Network, virtual LAN, VLAN Bridge

Contents

Proposed text for
Local and Metropolitan Networks —

Bridges and Bridged Networks

Amendment: Enhancements to Cyclic Queuing and Forwarding

(Amendment to IEEE Std 802.1Q™–2022 as amended by IEEE Std 802.1Qcz™–2022)

NOTE—The editing instructions contained in this amendment define how to merge the material contained therein into the existing base standard and its amendments to form the comprehensive standard.

The editing instructions are shown in ***bold italics***. Four editing instructions are used: change, delete, insert, and replace. ***Change*** is used to make corrections in existing text or tables. The editing instruction specifies the location of the change and describes what is being changed by using ~~strikethrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Deletions and insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this note will not be carried over into future editions because the changes will be incorporated into the base standard.

1. Overview

1.3 Introduction

Insert the following text at the end of 1.3 and reletter accordingly:

This amendment specifies procedures, protocols and managed objects for Enhanced Cyclic Queuing and Forwarding (ECQF). To this end, it

a) Specifies a transmission selection procedure that organizes frames in a traffic class output queue into logical bins that are output in strict rotation at a constant frequency;
b) Specifies a procedure for storing received frames into bins based on the time of reception of the frame.
c) Specifies a procedure for storing received frames into bins based on per-flow octet counters.
d) Specifies a protocol for determining the phase relationship between a transmitter's and a receiver's bin boundaries in time.
e) Provides managed objects, Management Information Base (MIB), and YANG modules for controlling these procedures.
f) Provides an informative annex to provide guidance for applying these procedures.

2. Normative references

Insert the following items into the list of Normative References:

3. Definitions

Insert the following definitions in the appropriate collating sequence, renumbering accordingly:

4. Abbreviations

Insert the following acronym(s) and abbreviation(s), in the appropriate collating sequence:

ECQF	Enhanced Cyclic Queuing and Forwarding

5. Conformance

5.4 VLAN Bridge component requirements

5.4.1 VLAN Bridge component options

Insert the following three sections at the end of 5.4.1 and renumber accordingly:

5.4.1.12 Extended Cyclic Queuing and Forwarding (ECQF) requirements

A VLAN Bridge component implementation that conforms to the provisions of this standard for ECQF (see Annex NF) shall

a) Support the ATS transmission selection algorithm as specified in 8.6.8.5.
b) Support the ATS scheduler state machines as specified in 8.6.11.
c) Support the requirements for Per-Stream Filtering and Policing (PSFP) as stated in 5.4.1.8.
d) Support the management entities for ECQF as specified in §12.TBD.
e) Support the management entities for PSFP as specified in 12.31.

5.4.1.13 Time-based ECQF requirements

A VLAN Bridge component implementation that conforms to the provisions of this standard for time-based ECQF (see Annex NF) shall

a) Support ECQF as stated in 5.4.1.12.
b) Support time-based ECQF bin selection as specified in §8.TBD.
c) Support the management entities for time-based ECQF as specified in §12.TBD.

5.4.1.14 Count-based ECQF requirements

A VLAN Bridge component implementation that conforms to the provisions of this standard for count-based ECQF (see Annex NF) shall

a) Support ECQF as stated in 5.4.1.12.
b) Support count-based ECQF bin selection as specified in §8.TBD.
c) Support the management entities for count-based ECQF as specified in §12.TBD.

5.13 MAC Bridge component requirements

5.13.1 MAC Bridge component options

Insert the following three sections at the end of 5.13.1 and renumber accordingly:

5.13.1.4 Extended Cyclic Queuing and Forwarding (ECQF) requirements

A MAC Bridge component implementation that conforms to the provisions of this standard for ECQF (see Annex NF) shall

a) Support the ATS transmission selection algorithm as specified in 8.6.8.5.
b) Support the ATS scheduler state machines as specified in 8.6.11.
c) Support the requirements for Per-Stream Filtering and Policing (PSFP) as stated in 5.4.1.8.
d) Support the management entities for ECQF as specified in §12.TBD.
e) Support the management entities for PSFP as specified in 12.31.

5.13.1.5 Time-based ECQF requirements

A MAC Bridge component implementation that conforms to the provisions of this standard for time-based ECQF (see Annex NF) shall

a) Support ECQF as stated in 5.13.1.4.
b) Support time-based ECQF bin selection as specified in §8.TBD.
c) Support the management entities for time-based ECQF as specified in §12.TBD.

5.13.1.6 Count-based ECQF requirements

A MAC Bridge component implementation that conforms to the provisions of this standard for count-based ECQF (see Annex NF) shall

a) Support ECQF as stated in 5.4.1.12.
b) Support count-based ECQF bin selection as specified in §8.TBD.
c) Support the management entities for count-based ECQF as specified in §12.TBD.

Insert the following section at the end of Clause 5 and renumber accordingly:

5.14 End station requirements for count-based ECQF

An end station implementation that conforms to the provisions of this standard for count-based ECQF (see Annex NF) shall

a) Support the ATS transmission selection algorithm as specified in 8.6.8.5.
b) Support the ATS scheduler state machines as specified in 8.6.11.
c) Support the requirements for Per-Stream Filtering and Policing (PSFP) as stated in 5.4.1.8.
d) Support the management entities for ECQF as specified in §12.TBD.
e) Support the management entities for PSFP as specified in 12.31.
f) Support count-based ECQF bin selection as specified in §8.TBD.
g) Support the management entities for count-based ECQF as specified in §12.TBD.

Insert the following Clause:

99. Receiver Phase Alignment Protocol

<< This section will describe the protocol presented in https://www.ieee802.org/1/files/public/docs2021/new-finn-CQF-sync-method-1121-v1.pdf. >>

Annex A

(normative)

PICS proforma—Bridge implementations[1]

<< No changes so far to this Annex, hence it is not shown in this draft. >>

1

Annex T

(informative)

Cyclic queuing and forwarding[2]

<< No changes so far to this Annex, hence it is not shown in this draft. >>

[2] In early discussions, CQF was known as the “Peristaltic Shaper” [B71].

Insert the following Annex:

Annex NF

(informative)

Enhanced cyclic queuing and forwarding

NF.1 Overview of ECQF

<< Some of the information in this annex will probably become normative, rather than informative. Some of it may ot be necessary for the final document. As of this draft, the editor's object is to collect information that may be needed for the final document. >>

<< Editor's note: This Annex text is largely taken from https://www.ieee802.org/1/files/public/docs2021/new-finn-multiple-CQF-0921-v02.pdf. It has not yet been cast completely into the mold presented in https://www.ieee802.org/1/files/public/docs2022/dv-finn-ECQF-annex-1122-v01.pdf. In particular, it needs to be reworked to have a clean split between output bin rotation, count-based input bin assignment, and time-based output bin assignment. >>

Enhanced Cyclic Queuing and Forwarding (ECQF) utilizes output queues (§Clause 6 and 8) divided into fixed-sized bins, which are made eligible for transmission in a circular fashion at a constant frequency. One queue operates at one frequency. Multiple queues on the same port can operate at different frequencies. Received frames are assigned to bins based on either time of arrival (Time-Based ECQF, NF.2) or per-Stream state machines (Count-Based ECQF, NF.3).

NF.4 discusses Stream Aggregation, which can reduce end-to-end latency of ECQF Streams and reduce the resources required to support ECQF. NF.5 suggests a number of further augmentations that can be made to ECQF to improve bandwidth utilization and worst-case latency.

This Annex assumes the reader is reasonably familiar with Cyclic Queuing and Forwarding (CQF) as described in Annex T. The frame timestamps used in this paper are described in Clause 90 of IEEE Std 802.3-2018. Preemption, or interspersed express traffic, is described in §6.7.2 and in Clause 99 of IEEE Std 802.3-2018.

NF.1.1 ECQF vs. CQF

The differences between CQF, as defined in Annex T, and ECQF, as defined here, are:

a) For a given value of the cycle time T_C on a given output port, Annex T provides two bins, each implemented as a separate class of service queue. Transmission gates (§8.6.9) are configured to enable the two queues to output alternately, each given time T_C to drain.

   ECQF allows two or more bins per output port per cycle time. It is not practical to dedicate one of only eight available class of service queues for each of these bins. One class of service queue can any number of ECQF bins for a single cycle time.

b) Annex T assumes that one value of T_C is sufficient for any given output port.

   ECQF allows multiple values of T_C, one for each ECQF class of service. The output cycles are constrained, on any given port, so that an integral, and never a fractional, number of shorter cycles are contained within any given longer cycle. We will assume that one class of service queue serves a single value of T_C.

c) Annex T uses synchronized time so that every output port in a TSN network switches bins simultaneously.

ECQF allows each Bridge to perform its bin switching at different times, subject to the above constraint b).

d) Annex T uses the same cycle time and phasing for the Stream gates (§8.6.5.4) as for the transmission gates (§8.6.8.4). The Stream gates select in which of the two output queues on a given port to store a received frame.

ECQF runs the same cycle time for input and transmission gates, but adjusts the phase of the Stream gates on a port to match the phase of the frames arriving from the transmission gates of the Bridge transmitting to that receiving port.

e) Annex T assigns all incoming frames to the only CQF queue that is open on any given port.

ECQF uses either Time-based or Count-based methods for assigning frames to output bins.

NF.2 Time-based ECQF

CQF (Annex T) assumes that every Bridge has a system clock that is synchronized with the other Bridges' system clocks in the network, so that configuring a transmission time in one Bridge has meaning in another Bridge. ECQF does not require synchronization of the system clocks, but does require frequency lock. That is, there is a maximum difference in the elapsed time between two events as measured by two Bridges' system clocks, no matter the length of that elapsed time.

NF.2.1 Timeline for time-based bin assignment

We have two nodes, A and B. Both are running an instance of ECQF on each of multiple ports, more-or-less as described in Annex T.

After a bin-swap event on a particular port, node A transmits all of the frames in one bin towards receiving node B, not necessarily in a single burst. After some gap following the transmission of the last frame in the bin, another bin-swap event is performed. At this point, it starts transmitting the frames from the next bin. The bin-swap 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 bin such that 1) frames that were in the same bin in node A, and are transmitted on the same port from node B, are placed into the same bin in node B; and 2) frames in different bins in node A are placed in different bins in node B.

Figure NF-1 shows an example of ECQF. 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 NF-1, we use the following notation for time intervals:

T_C	nominal (intended) period of the bin-swapping cycle
T_I	maximum interference from lower-priority queues, one frame or one preemption 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 added to Stream data if this traffic class is preemptable
T_D	end-of-cycle dead time optionally imposed on node A by node B
T_W	wait time during which the bin is neither receiving nor transmitting frames
T_{AB}	effective phase difference between cycle start times for input from A and output from B

For time-based ECQF, T_{AB} must remain constant; that is, any variation in is included in T_V. Bounding this variation is another way of saying that all Bridges' T_C values are exactly equal.

Following the definitions of transmission gates in §8.6.8.4, the red ticks in timelines 1 and 4 in Figure NF-1 represent the earliest possible moment at which the first bit of the destination address of the first frame of the

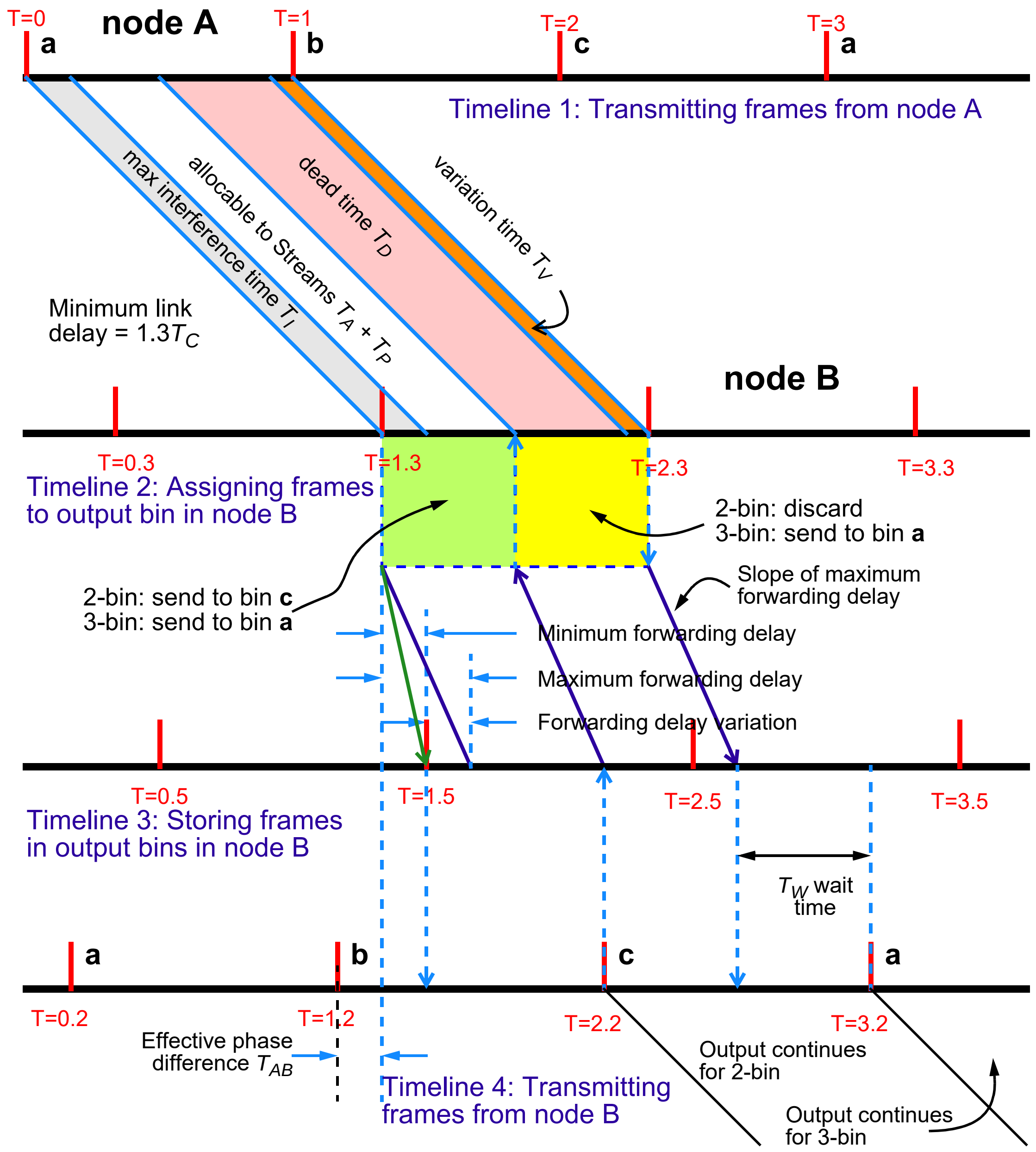
T=0
a
node A
T=1
b
T=2
c
T=3
a
Timeline 1: Transmitting frames from node A
max interference time T_I
allocable to Streams $T_A + T_P$
dead time T_D
variation time T_V
Minimum link delay = 1.3T_C
node B
T=0.3
T=1.3
T=2.3
T=3.3
Timeline 2: Assigning frames to output bin in node B
2-bin: discard
3-bin: send to bin **a**
Slope of maximum forwarding delay
2-bin: send to bin **c**
3-bin: send to bin **a**
Minimum forwarding delay
Maximum forwarding delay
Forwarding delay variation
T=0.5
T=1.5
T=2.5
T=3.5
Timeline 3: Storing frames in output bins in node B
T_W wait time
a
b
c
a
T=0.2
T=1.2
T=2.2
T=3.2
Effective phase difference T_{AB}
Output continues for 2-bin
Timeline 4: Transmitting frames from node B
Output continues for 3-bin

Figure NF-1—Example of timelines for time-based ECQF

cycle can be transmitted. These ticks are ultimately driven by the frequency-locked clock. They are the basis for all bin transmissions. If Enhancements for Scheduled Traffic (ETS, §8.6.8.4) are used for controlling the output bins, the ticks are the points in time when the transmission gate of one queue is closed, and the next queue's transmission gate is opened. These are the points in time as programmed into the managed objects that control ETS. An implementation may need to schedule open and close events in anticipation of the time specified in the managed objects in order to maximize throughput. Note that the preamble of an IEEE Std 802.3 Ethernet frame can be transmitted before a gate open event.

NF.2.1.1 Output timeline 1

Figure NF-1 shows an interference delay T_I (the gray area) between node A's bin-swap event (the red ticks in Figure NF-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 NF-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 an ECQF Stream. This value is always bounded. See NF.2.5.

There can be some variation in the time from the selection of a frame for output in node A to the timestamp moment, when the first bit of the destination MAC address is transmitted (see Clause 90 of IEEE Std 802.3-2018). 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 nodes' 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 Std 802.3 transmit and receive timestamps and the local clock that drives the bin-swap 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 in NF.2.2, the next hop can impose a dead time T_D on this hop. This is a time at the end of the cycle, during which no frames can be transmitted from the bin, 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 an output bin together, at line rate, with no interference from lower-priority queues on the same output port. (Interference from higher-priority queues is described in NF.2.5.) Given that is true, then the total time per cycle that can be used for transmitting Streams is:

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

This T_A is a maximum, local to a particular class of service and 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 transmission 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_D and parts of T_V. T_D is supplied by configuration, or by the node to which the output port is connected. 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 transmission gates are defined in §8.6.8.4.

NF.2.1.2 Receive timeline 2

The timeline at the receiving port is timeline 2 in Figure NF-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.

On timeline 2, each frame is assigned to a bin on an output port based on the timestamp (Clause 90 of IEEE Std 802.3-2018) on the 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 NF-1. It is clear from the figure that T_{AB} must be known in order to compute T_D 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, e.g. that described in Clause 99.

Once T_{AB} is known, all of the timing relationships shown in Figure NF-1 can be computed. The phasing of the nodes' output bin cycles affects the end-to-end latency of any Stream, so that phasing must be known when the end-to-end latency is computed. However, the end-to-end latency is not necessarily an integer multiple of the cycle time. One could even adjust the phasing (by adjusting the phase of timeline 4) to favor certain paths through the network.

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

If a frame (belonging to a Stream) is received that straddles a cycle (first bit in one cycle on timeline 2 of Figure NF-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, unless the frame is discarded or marked down to best-effort service, it can cause disruption of delivery guarantees farther along in the network.

NF.2.1.3 Storing frames timeline 3

The timeline at the point where frames are stored into an output bin is timeline 3 in Figure NF-1. The red ticks on timeline 3 mark the earliest point at which the first frame transmitted from a particular bin could reach the output bins (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 NF-1 include the time to install the frame in the output bin and for its presence to filter through to the point that it can be selected for output.

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

Note that an implementation may require a minimum offset between timeline 3 and timeline 4. That is, a time lag may be required between the last opportunity to store a frame in a bin, and the earliest time at which the first bit of a frame from that bin can appear on the link. Some time could, for example, be necessary in order schedule the transmission of frames across multiple queues in order to ensure that the requirements of strict frame priority and back-to-back frame transmission (NF.2.11) can be met.

NF.2.1.4 Transmitting frames timeline 4

Depending on whether 2-bin or 3-bin mode is used, one can trade off reduced total available bandwidth against per-hop delay. timeline 4 in Figure NF-1 shows the two options for the choice of which output cycle in node B is used to transmit frames that were transmitted from bin a in node A.

NF.2.2 Calculation of dead time T_D

Timeline 3 in Figure NF-1 shows the calculation of T_D, which applies only to 2-bin mode. The starting point of T_D is the moment that the output cycle starts (the tick on timeline 4), moved backward by the worst-case forwarding delay. This is the last moment on timeline 3 that a frame can be assigned to bin c in the example in Figure NF-1. The end of T_D is the end of the cycle T_C, less the variation time T_V. In 3-bin mode, T_D is zero.

T_D can only be computed by 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. MSRP (Clause §35), then the value of T_D 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:

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

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

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

NF.2.3 More than 3 output bins

The discussion over Figure NF-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 bins, and assign received frames to bins whose output is scheduled far enough ahead in time to ensure that, in the worst case, they will arrive in the bin before the bin begins transmitting. This works only because the bin 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 (see C.9 of IEEE Std 802.1CB-2020). In this case, more than 3 output bins can be allocated, and used to impose a delay of an arbitrary number of cycle times T_C on every frame.

Each output port in a node, and each output port along the path of a Stream, can have a different number of bins, whether 2, 3, or 50. Furthermore, one Stream can use (e.g.) 3 bins on an output port, while another Stream, which needs a path-matching delay, can use 12 bins on the same port. (Of course, this would require per-Stream configuration.)

NF.2.4 Multiple T_C model

With CQF as it is described in Annex T, we are limited to a single class of service (a single value of T_C) and to 2-bin operation, only. We have already discussed 3-bin (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, because all frames for all Streams sharing a port 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 NF-2.

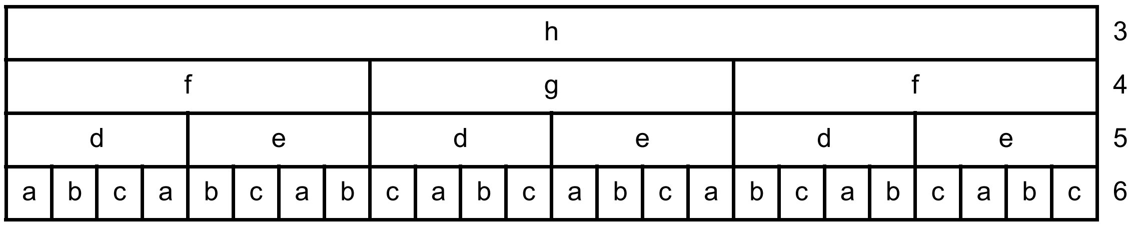

Figure NF-2—Multiple T_C values

In Figure NF-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 bins 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 NF-2 label which bin is output during the cycle. There are 9 bins a through i. Bin i, the second bin at priority 3, is not shown. In this example, priority 6 uses three bins, because the timing is tight; the others use two each.

We assume here that the receiver of a frame can identify the particular ECQF instance (T_C value) to which the frame belongs by inspecting the frame. A TSN Bridge could use the L2 priority field of a VLAN tag, or it could use the DSCP field of an IP packet. IEEE Std 802.1CB provides 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 NF-3, where we illustrate the timing of transmission of frames from three levels of ECQF and the best-effort (BE) level. Note that ECQF 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.

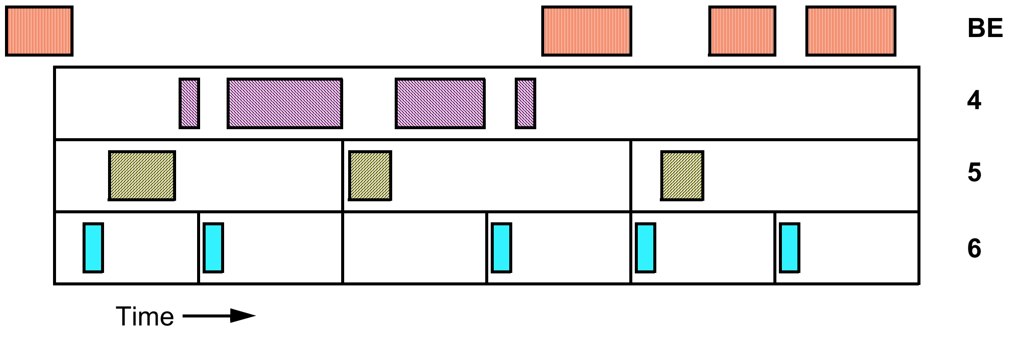

Figure NF-3—Transmission timing

For both CQF and ECQF, a given Stream is allocated a fixed number of bits that it can transmit per cycle T_{Cn}. A scheduler would typically assign each Stream to the highest-numbered (fastest) ECQF instance such that, at the Stream's bandwidth and frame size, the Stream occupies some space in every bin at that level. Then, ECQF will maintain one or two frames in its bins 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 bins must be used for bins h and i. If all Streams are fast and are allocated to T_{C6}, then only three small bins are used—bins a, b, and c are rapidly re-used.

NOTE—There are many ways to allocate buffer space to individual frames. Running ECQF at 5 levels does not increase the bin memory requirements beyond that of 1-level ECQF. 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 bin 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 one frame per cycle, with no overprovisioning, 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.

NF.2.5 Preemption and interference

Frame preemption is described in §6.7.2 and in Clause 99 of IEEE Std 802.3-2018. 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 NF.2 and Figure NF-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 Std 802.3). The interference time is shown as a gray parallelogram attached to timeline 1 in Figure NF-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 NF-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 during the cycle.) 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.

NF.2.6 T_C computation

We can also compute a suitable value for T_C, given a desired value for T_A:

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

Annex T CQF assumes the 2-bin scheme, and so assumes that T_D 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_D can be eliminated by using the 3-bin scheme of ECQF.
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.

NF.2.7 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 NF-4.

In Figure NF-4, we have allocated 40% of the link bandwidth to the Streams using priority 5, and 50% of the link bandwidth to the Streams using priority 4. 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 (see NF.5.4), we can get the situation shown, in the shaded bins. Bins b5, a5, and then again, b5 emit their frames (at high priority) at the indicated times. Even though the priority 5 Streams take up only 40% of each level-2 cycle, they can output 6 frames over the course of cycle b4, thus taking up 60% of the bandwidth during that period. There is, therefore, 110% of the bandwidth that must be output during the

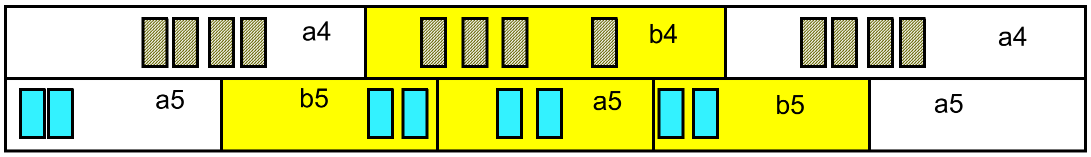

Figure NF-4—Variable T_C

period that b4 is transmitting. b4 cannot output all of its data. Some of it must be somehow delayed, but there is no place to put that data. Deterministic QoS is not obtained.

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 NF-4 is avoided. It also bounds the number of preemption events that can steal bandwidth from a given priority level.

NF.2.8 Basic requirement for determinism

ECQF guarantees the Deterministic QoS by the following argument.

We assume that the Talker uses ECQF. Non-ECQF inputs to a Bridge are discussed in NF.3.1.

We consider only one value of T_C along the path of a given Stream from Talker to Listener. NF.2.8.1 and NF.2.8.2 deal with exceptions to this assumption.

The contract between the Talker and the network is in terms of 1) a maximum frame size, and 2) a maximum number of bit-times on the medium per cycle time. For Ethernet, the number of bit times for a given frame is equal to (the frame size from destination MAC address through Frame Check Sequence, plus 20 bytes for preamble and inter-frame gap) times 8 bits per byte.

A number of considerations reduce the fraction of the total time T_C that can actually be used to transmit data. See NF.2 for details. For example, the maximum frame size of each Stream allows us to determine the worst-case interference that a given Stream can have on higher-priority Streams. All of these considerations are bounded; if an implementation cannot bound one or more of these considerations, then it cannot guarantee the Deterministic QoS in a ECQF network.

In a detailed timing analysis, we will note that the first bit of the MAC address of a frame is never transmitted before the start of the window time (according to the local time in the transmitter) and the last bit of the interframe gap (always) and the preamble of the next frame (if any) are is transmitted before the end of the window.

In order to obtain Deterministic QoS for each Stream, we must ensure that no bin is ever asked to hold more data than it can transmit during one cycle time T_C. Since the amount of data supplied by any given Stream in one cycle is set by contract, we can accomplish this as follows:

a) The Talker contract is enforced when a Talker's frames are first placed into a ECQF output bin after entry to the network. That is, the frames from a given Stream do not exceed the Talker contract in the first ECQF output bin in the network.

   Ingress conditioning and/or policing is discussed in 7.1. NWF

b) Frames belonging to the same Stream that are in the same ECQF output bin in one Bridge in the network are placed in the same ECQF output bin in all subsequent Bridges along a shared path.

NF.2, and particularly Figure NF-1, show the details of how this is accomplished. The key is to get the Stream gates synchronized with the transmission gates of the transmitting system, offset by the link delay. Frames received during one input cycle are always placed in the same bin. If the input cycle is synchronized with the previous hop's output cycle, then cycle integrity is maintained. (Of course, this only works for point-to-point links.)

c) There is no fan-in for a particular Stream.

We assume that the path of a Stream reservation through the network is known and does not change. A given Stream enters a Bridge through one port only, although it may be a multicast Stream, and thus be enqueued and transmitted on more than one port.

d) Admission control ensures that, on any given output port and cycle time T_C, the total bits times for all Streams passing through that port and T_C value does not exceed the available transmission time on that port. (This assumes that no Bridge has a limitation on available receive time on an input port that is smaller than the attached output port's available transmit time. The implications of such a limitation are obvious.)

NF.2.8.1 4Admission control for multiple T_C values

NF.2.4 describes the operation of ECQF with multiple T_C values operating simultaneously on one output port. Figure NF-3 shows an example of a sequence of transmissions. We observe that the shortest cycle times operate at the highest priority, and the longest at the lowest priority. Because different ECQF priority levels may have different maximum frame sizes, and because some may enable preemption, different priority levels may have different amounts of time during one cycle that cannot be allocated to Stream transmission. Clearly, allocating time for any ECQF priority level reduces the time allocable to other priority levels; there is only one physical link.

An administrator may wish to restrict allocation of ECQF transmission times to leave room for transmitting non-ECQF frames, either best-effort traffic or other, lower-priority TSN traffic.

For a new Stream to be admitted, it must be true that the available transmission times over all of the ECQF levels on all of the output ports through which the Stream travels have not been exhausted. At any given ECQF priority level x, one can add the bits allocated to all Streams in one cycle at ECQF priority level x, plus the sum over all more-important ECQF priority levels y (faster cycles), of the product of the number of bits per cycle allocated at that level times the number of cycles at that level contained within one cycle at level x. At every level, the total must not exceed the maximum number of allocable bits at that level.

(This calculation is simpler if, at every ECQF priority level, there is the same percentage of dead time and slop for inaccuracies, but this is not necessarily the case.)

NF.2.8.2 Changing T_C values along the path of Stream

If a Stream enters a Bridge using a cycle time T_C, and is being transmitted on an output port with cycle time n*T_C, then n successive input cycles can be deposited in the same output bin with no problem, as long as the larger cycle time's dead time requirements are met. (This is not a trivial exception, as the larger cycle's dead time occurs at the end of the large cycle, and thus may take up much or even all of one small cycle.) Equivalently, the input port can be configured with the slower cycle time to match the output port in the same system. Of course, when making the reservation for that Stream, the adjustment of its contract must be made; it is allocated n times the number of bits in the slower cycle than in the faster cycle.

In all other cases, when a Stream changes cycle times, the Stream must pass through a conditioning step, such as a count-based ECQF step (see NF.3.1), to ensure that the Stream never exceeds its contract in the new cycle time.

NF.2.9 Computing the actual end-to-end latency for time-based ECQF

After adjusting to get the receiving window aligned with the previous-hop transmitting window, a Bridge knows the “effective phase difference T_{AB}” described in NF.2. Referring to Figure NF-1, this allows the Bridge to compute the difference, in time, between the start of an input window for the Stream, and the start of the output window in which a frame received in that input window will be transmitted. This is the dwell time for the frame in this Bridge. Adding this to the one-way link delay gives the per-hop delay for frames in the Stream. At egress from the network, there is a margin of one cycle time less one frame transmission time for delivery of the frame, as the frame can be transmitted at any point during the cycle, but must both start and finish its transmission within the cycle. The delay at ingress is somewhat more complicated to measure, as it depends upon the method used by the Talker and the ingress Bridge to shape its transmissions.

If we look again at Figure NF-1, we can see that the difference between using two and three bins for a given input-output port pair is really a matter of rounding up the link delay to an integral number of cycle times. If the sum of link delay and phase delay between output cycles is negligible, or happens to be very nearly an integer multiple of the cycle time, then the yellow “discard” area is small, and two bins can be used. If sum is larger, then one necessarily chooses between a smaller allocation (large discard area) and increased delay.

NF.2.10 Output bin selection

The minimum number of bins required (usually 2 or 3) depends on the relative phase of the input and the output cycle start times. But different input ports generally will have different phases. Thus, the number of bins used by any given output port will vary with the input port; an output port can have three bins, for example, but for some input ports, there are never frames from that port in more than two bins.

For the present time, we will assume the following method for receiving a frame and assigning it to a bin. It is important to stress that there are many ways to accomplish the same task.

Let B_o be the number of physical output bins on port o. We compute N, the least common multiple over all B_o in the system. Each input port i assigns each received frame a bin selector S, which is an integer in the range 0 through N—1, and which increments (modulo N) each input cycle. Thus, frames transmitted from the same bin are assigned the same S value at the receiving end of the link.

At the output port o, each of the B_o bins is identified by a bin number in the range 0 through B_o—1. A variable X_o indicates which bin is currently transmitting. X_o increments once modulo B_o each output cycle.

When a frame arrives at an output port, it is assigned to a bin b using the formula:

$b = (S + P_{io}) \bmod B_n$

Where P_{io} is the cycle phase offset from input port i to output port o and B_n is the number of output bins on the port. See NF.2.12.4 for the determination of P_{io}. Note that in the extreme case of all output ports using two bins, all synchronized, and all input cycles in phase with the output cycles, the table P_{io} reduces to a single value, 0 or 1, and we have ECQF from Annex T.

It is desirable in some cases to deliberately use more bins than are required for insurance against congestion loss in order to match the end-to-end delay of a Stream across different paths through the network. If such delay matching is performed per-Stream, instead of per-input port, then per-Stream P_{io} values are required for bin selection.

P_{io} is not dynamic, though its values may change when the relative phasing between an input port cycle and the transmitter feeding it change suddenly. Such a change will always disrupt the ECQF service guarantees.

NF.2.11 Implementation requirements

The admission control calculations presented here depend upon the transmitting port being able to select the correct frame to transmit according to strict priority among the ECQF priority levels, and initiate all transmissions in that order, without introducing extra inter-frame gap time. Since, with ECQF, no bin has frames both arriving and being transmitting at the same instant, this should pose no insurmountable problems for implementors.

Delivering the deterministic QoS using ECQF depends upon a transmitting port being able to select the correct frame to transmit according to strict priority among the ECQF priority levels, and initiate all transmissions in that order, without introducing extra inter-frame gap time. Since, with ECQF, no bin has frames both arriving and being transmitted at the same instant, this should pose no insurmountable problems for implementors.

NF.2.12 Parameterization of time-based ECQF

Let us go through the exercise of initializing an input/output port pair for ECQF. In the process, we will collect a set of parameters that can be used with protocols and/or network management to monitor and control the operation of ECQF.

NF.2.12.1 Cycle wander

Adjacent Bridges must be frequency locked as described in NF.2. For any given port, there is a worst-case system clock difference, sysClockVar, between this Bridge's system clock and the neighbor system attached to the port. Its units are a time difference. We will assume that this parameter is configured by management, based on network design parameters and system data sheets. It is possible that this parameter can be adjusted during network operation. A Bridge could have more than one system clock, and be connected to another system by multiple links, but there is only one value for sysClockVar for any given port, because we assume point-to-point links. We will assume that the variation can be in either direction, this-end-late or this-end-early.

Stream gates and transmission gates can operate under control of a clock that is local to a port. The management controls that configure the schedule are defined in terms of the system clock. The Bridge aligns the port clock(s) with the system clock either periodically or continuously. There is thus a worst-case excursion of the actual start of a cycle from the time configured in terms of the system clock. We parameterize this with four parameters for each port, cycleInMaxEarly, cycleInMaxLate, cycleOutMaxEarly, and cycleOutMaxLate, all measures of time. cycleIn*** is for the Stream gate error, cycleOut*** for the transmission error. ***Early is the worst-case for starting the cycle before the system clock time, and ***Late the worst-case for starting after the system clock time. Having both early and late parameters allows for different implementation methods for aligning the port schedule to the system clock. This parameter is computed by the Bridge from knowledge of the implementation, and is constant over the lifetime of a physical connection.

NF.2.12.2 Link delay variation

The time taken for a frame to travel from the transmitter to the receiver can vary for two reasons: the actual delay can change, due for example to temperature variations in a multi-kilometer link, and the measurement of the link delay can vary due to various clock inaccuracies. We will deal only with actual variations, not measurement variations. 1-21-0056-00-ICne-input-synchronization-for-cyclic-queueing-and-forwarding explains why this is possible.

NF.2.12.3 Calculating the number of bins required

The procedure to calculate the number of bins needed on an output port to support one particular input port is as follows:

a) Establish a Nominal Input Cycle Start time (NICS) for the input port, and a Nominal Output Cycle Start time (NOCS) for the output port. The NICS and NOCS each repeat every T_C seconds, according to the system clock. We will assume that the offset between them is a constant (i.e., they are both driven by the same system clock).

b) Compute the earliest time, relative to the NICS, at which the first frame of a cycle can receive its IEEE Std 802.3 clause 90 timestamp. This frame is assumed to be a minimum-length frame (64 bytes plus overhead).

c) Compute the earliest time, relative to the NICS, at which a bin on the output port must be eligible to receive the frame. This is equal to the timestamp time in bullet b) plus the minimum time required to move the frame through the Bridge to the output bin.

d) Compute the latest time, relative to the NICS, at which the last frame of a cycle can receive its timestamp. This frame is assumed to be a minimum-length frame.

e) If the difference between the earliest timestamp and the latest timestamp is greater than or equal to the cycle time T_C, then dead time must be imposed on the transmitter, at the end of the cycle, to reduce the difference.

f) Compute the latest time, relative to the NICS, at which the last frame of a cycle can be stored into an output bin and be ready for selection for transmission, given the worst-case forwarding delay through the Bridge.

g) Convert these earliest b) and latest d) arrival times to times relative to the NOCS of the output port.

h) Arbitrarily label an input port NICS event NICS0. Determine the latest subsequent NOCS event, which we will label NOCS0, during which the earliest-arriving frame of NICS0 must be stored in the output queue.

i) Determine the earliest subsequent NOCS event, which we will label NOCSn, before which the latest-arriving frame from NICS0 can be stored in the queue, and still be available for transmission at the start of cycle NOCSn.

j) The number of cycles NOCS0 through NOCSn, inclusive, is the number of bins required for the input/output port pair, B_{io}.

The number of bins required can sometimes be reduced by:

— Imposing a larger dead time on the transmitter feeding the input port, at the end of every cycle;

— Altering the phase of the output port’s cycle; and/or

— Imposing implementation-specific limitations on the Streams, e.g. reducing fan-in to an output port, or restricting bridging/routing features to reduce forwarding delay variation.

Finally, let us observe that large link delay variations can be accommodated by varying the above calculation. Assuming that the variations take place slowly, and that changes in relative phase between transmitter and receiver are detected using a protocol (e.g. that in Clause 99), the difference between the maximum and minimum link delay can be added to the difference between the earliest- and latest- arriving frames to increase the number of bins allocated. The phase of the Stream gate can be altered by small increments as the protocol detects the phase differences, without gaining or losing cycles in the transfer. Of course, the maximum adjustment made per phase adjustment event must be removed from the allocable bandwidth.

NF.2.12.4 Initial bin phase

The number of bins required on an output port is the maximum required over all input ports. This may be further increased by intentional delays (NF.2.3). When initializing an input port, a correspondence must be made between the input and output ports, so that a frame received on the input port will be stored in a particular bin in the output port, the one that will become the transmitting bin in the appropriate number of output cycles in the future.

The phasing between input and output ports' cycles, and thus the number of bins in port *o* used by port *i*, is determined by the P_{io} table defined in NF.2.10. We compute P_{io} when initializing ECQF, or when the relative phase of the input and output ports change significantly, by selecting a time T that coincides with the start of an input cycle on input port *i* and computing:

$P_{io} = (X_o — S_i — B_{io} + 1) \bmod N$

Where X_o is the identity of the transmitting bin on output port *o* at time T, B_{io} is the total number of bins required of output port *o* by input port *i* (including the transmit bin), S_i is the value of bin ID *S* assigned by port *i* during the input cycle starting at time T, and *N* is the range of S_i, the least common multiple of the number of physical bins over all output ports.

NF.2.13 Externally-visible ECQF managed objects and protocol items

The following list includes both objects of interest to a network manager, and information elements that might be exchanged using a link-local protocol. Most items could be carried in a protocol as a check on proper configuration of adjacent ports, with varying degrees of utility for different items. Some items can only be computed by one system, and must also be known to the adjacent system. It is for further study what protocols would be used for such information transfers, or and/or whether the transfers are best accomplished using network management.

NF.2.13.1 Cycle and priority structure managed objects

For each output port and each input port, separately, we have:

a) The cycle time of the slowest ECQF priority value.
b) The priority value of the slowest ECQF cycle.

For each priority level running ECQF on an input port or an output port (separately), we have:

c) The layer 2 priority value
d) The number of cycles at this priority level contained within one next-lower priority value cycle.

There are other, equivalent, ways to formulate this same information. We can divorce layer 2 priority code point from importance, for example.

These parameters are not expected to change over the lifetime of a data Stream. A system would not be expected to obtain this configuration information from a neighbor through an ECQF-specific protocol, though exchanging this information could be done to discover of configuration errors.

NF.2.13.2 Cycle phase managed objects

For each output port and input port, separately, we have:

a) The start time of the slowest ECQF cycle, in terms of the system clock.

This variable establishes the phase of the input or output cycle (NICS and NOCS, see NF.2.12.3). Typically, this variable would be managed the network administrator for output ports. For time-synchronized systems, it can be administered for input ports, as well. Alternatively, the input phase can be determined dynamically, and be read by the network administrator. Because it is in terms of the system clock, it is of no interest to neighboring systems except, perhaps, as a configuration error check for time-synchronized networks.

NF.2.13.3 Cycle variation information

For each output port only, we have:

a) The largest offset from the nominal (system clock) NOCS event to the actual cycle start time, in the negative (actual earlier than NOCS) direction.
b) The largest offset from the nominal (system clock) NOCS event to the actual cycle start time, in the positive (actual later than NOCS) direction.

There are other ways to express the information in these two items. These values must be known to the connected input port in order for that system to compute its buffer space and dead time requirements (cycleOutMaxEarly, cycleOutMaxLate in NF.2.12.1). This information transfer could be accomplished by means of a protocol, managed objects, or by restrictions on implementations.

NF.2.13.4 Dead time / bandwidth balance information

There remains the balancing of conflicting goals between dead the percentage of a cycle that is available to transmit critical data Streams, and the number of bins required on the output port. Increasing the dead time can reduce the number of bins required, and thus the end-to-end latency of a data Stream, as described in NF.2.12.3. There are, at the very least, the following ways to make this decision:

— Configure the output cycle phase and number of bins to use for all Bridges, in order to establish a constant per-hop delay in a network with short links. Let each system compute the dead time on each input port required to make this work, and the bandwidth available for allocation. Convey the required dead time either by protocol or by management to the transmitters, and the available bandwidth to the admission control system.
— Configure the output cycle phase on all Bridges. Configure minimum and maximum allocable bandwidth values for each ECQF priority level. Let each system compute the minimum number of bins required to meet the minimum bandwidth value, taking advantage of the maximum bandwidth value to compute a dead time value that minimizes the number of bins required. This would be useful in a network with very long links. Convey the resultant dead time to the transmitter via protocol, and the resultant allocable bandwidth to the admission control system.
— Using data sheet information, configure all parameters via network management. Adjust the output port cycle phasing to optimize the delay for certain specific Streams.

Given that context, the following items are required by ECQF:

a) Per input port, per priority level, the total dead time that must be provided by the adjacent transmitter at the end of each transmit cycle.

   There is a component of this dead time computed in this section, as well as one computed in item (5) of NF.2.12.3. The sum of these must be known to the adjacent transmitting port.
b) Per output port, per priority level, the total dead time that is to be provided at the end of each transmit cycle.

   This can be configured, obtained from the adjacent input node, or be a maximum of these values.
c) The allocable bandwidth for this input port and priority level.

   This has three components, the minimum of the allocable bandwidth over all output ports reachable from this input port (in the input port's own system), any limitations imposed by the input port implementation, and any maximum imposed by management. Whether this is computed by, received by, or even known by the output port, or whether allocable bandwidth is the concern only of the admission control system, is an open question.
d) The allocable bandwidth for this output port and priority level.

   This can be configured, computed from the adjacent input node's requirements, or be a minimum of these values. . Whether this is computed by, received by, or even known by the output port, or whether allocable bandwidth is the concern only of the admission control system, is an open question.

NF.3 Count-based ECQF

As described in NF.2, time-based bin assignment assigns frames to output bins based on the time of arrival of the frame, and requires that the output queues of successive hops along an ECQF path run at exactly the same frequency, in order to ensure that no bin's capacity can be exceeded. This requirement can be relaxed, at the cost of implementing a state machine for each Stream passing through each output port. Then, the output queues along the path can run nearly the same frequency, and their relative phases (T_{AB} in Figure NF-1) can diverge.

Count-based bin assignment provides a counter state machine for each Stream that allows that Stream to store no more than its contracted amount of data per cycle into any given ECQF bin. Frames above that limit are stored in subsequent bins, up to the maximum amount of buffer space allowed that Stream, whereupon excess data is discarded.

NF.3.1 Using both count-based and time-based ECQF

A Stream entering a Bridge from a correctly-configured Bridge or end station that runs ECQF, and that has reserved bin space allocated for it, will not disrupt the deterministic behavior of ECQF. However, an ECQF Bridge could receive input from a Bridge, a Talker, a router, or any other device that uses some deterministic algorithm(s) to condition its Streams, but uses an algorithm other than ECQF. We assume that reservations (contracts) for these Streams can be translated into ECQF terms, with perhaps some overprovisioning required. Long term, the data adheres to the contract, and will not disrupt determinism. But, since the transmitter is not using ECQF, we cannot use just a frame's arrival time to assign it to a bin, except by overprovisioning ECQF sufficiently to accommodate the worst-case burst behavior of the algorithm employed by the sender. We would like to accommodate such input.

The count-based bin assignment makes this possible. A Bridge uses the same bin structure and output methods described for time-based bin assignment in NF.2, but instead of obtaining the bin selector S from the time of receipt of the input frame, as described in NF.2.10, it uses a state machine dedicated to each ECQF Stream using count-based bin assignment, to determine the bin selector. Extra bins are provided to accept such bursts. At the next hop, time-based ECQF can be used.

A given output bin can accept input from both time-based and count-based Streams, as long as they share the same cycle time; separate count-based and time-based bins or queues are not necessary. In addition, a paranoid network administrator could very well configure count-based bin assignment on every Stream in a frequency-locked network, in order to guard against misbehaving Bridges or Talkers. That is, while count-based bin assignment can be thought of as separate algorithm from time-based bin assignment, it can also be thought of as a protection mechanism for time-based assignment that can be employed as need, and when employed everywhere, removes the restriction that Bridges operate at exactly the same frequency.

In many networks, count-based and time-based bin assignment can be used at the same priority level in one network. The choice between count-based and time-based bin assignment can be made on a Bridge-by-Bridge basis, and not be visible to the Talker, the Listener, or the user.

NF.4 Stream Aggregation

<< As discussed in IEEE 802.1 TSN meetings, this section will be expanded significantly. >>

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 ECQF 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 aggregated[3] into large transmission units.)

A particular form of Stream aggregation is useful for both reducing the number of Streams tracked by a Bridge, and for improving the per-hop latency of a Stream. In this type of aggregation, a number of Streams are treated as a single Stream, with a single reservation, traversing a single path, for some portion of their journey through the network. It does not matter whether the frames are actually encapsulated in some common wrapper, or whether they are simply treated identically (e.g. given the same IEEE Std 802.1CB Stream_identifier).

What does matter for ECQF is that the aggregate Stream, which has a bandwidth equal to the sum of its component Streams, can be assigned a ECQF level with a faster T_C than its components could make use of. Less buffer space is used for the aggregate than for the separate Streams, because only one frame of the aggregate need fit in a cycle, instead of one frame per component Stream. The faster T_C lowers the latency and reduces the buffer requirements for all its component Streams.

In general, this requires that the aggregate Stream pass through a count-based bin assignment state machine when it is formed from its components, and that each component pass through a count-based bin assignment state machine if and when it is again separated as an individual Stream and passes, presumably, to a slower T_C value.

NF.5 Other issues

NF.5.1 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 4 000 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 13 000 bit times, which is approximately equal to a maximum-length Ethernet frame, and that the cycle time T_C = 100µs. 13 000/100µs = 130 Mbits/s. But, allocating a bandwidth of 13 000 bits/T_C will not give the Stream 130 Mb/s. In the worst case, one 13 000 bit frame followed by one minimum-length frame = 672 bits, the Stream gets (13 000+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 (13 000–8) wasted bits in each cycle. Therefore, we can guarantee 130 Mb/s using a cycle time of 500µs by provisioning (5*13 000

[3]Not to be confused with Link Aggregation

+ 13 000 – 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.

NF.5.2 Tailored bandwidth offerings

We can note that, in a service provider environment, overprovisioning can be almost eliminated by a combination of 1) Stream Aggregation (NF.4) 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.

In a service provider environment, overprovisioning can also be improved by 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. This way, the overprovisioning required for meeting an arbitrary distribution of requirements using a small set of T_C values is eliminated. (Or, at least, shifted to the customer's shoulders.)

NF.5.3 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.

NF.5.4 CQF and credit-based shaper

Looking at Figure NF-3, we see that, once the major cycle at priority level 4 begins transmitting, the best-effort traffic is interrupted until all of the ECQF level 4 data is transmitted. At some point, as the amount of traffic in a very slow ECQF cycle increases, the burstiness of the best-effort transmission opportunities could, in theory, become a problem. This can be mitigated by applying a credit-based shaper function to the slowest ECQF cycle(s). However, the parameters of this shaper must be adjusted as the load on the slow ECQF cycle(s) changes, because a Bridge must always finish transmitting all of the data in a bin. Thus, adding a credit-based shaper would detract from the most-significant advantage of ECQF—its freedom from requiring reconfiguring a Bridge each time a Stream is added.

Annex NG

(informative)

Bibliography

Insert the following references in the appropriate collating sequence and renumber accordingly:

[B1] Seaman, Mick, “Paternoster policing and scheduling” http://www.ieee802.org/1/files/public/docs2019/cr-seaman-paternoster-policing-scheduling-0519-v04.pdf,