P802.1Qdv/D0.4
November 14, 2023
(Amendment to IEEE Std 802.1Q-2022 as amended by IEEE Std 802.1Qcz, P802.1Qcw/D2.2, and P802.1Qcj/D2.6

Draft Standard for Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment: Enhancements to Cyclic Queuing and Forwarding

Prepared by the
Time-Sensitive Networking (TSN) Task Group of IEEE 802.1

Sponsor
LAN/MAN Standards Committee
of the
IEEE Computer Society

Important Notice

This document is an unapproved draft of a proposed IEEE Standard. IEEE hereby grants the named IEEE SA Working Group or Standards Committee Chair permission to distribute this document to participants in the receiving IEEE SA Working Group or Standards Committee, for purposes of review for IEEE standardization activities. No further use, reproduction, or distribution of this document is permitted without the express written permission of IEEE Standards Association (IEEE SA). Prior to any review or use of this draft standard, in part or in whole, by another standards development organization, permission must first be obtained from IEEE SA (stds-copyright@ieee.org). This page is included as the cover of this draft, and shall not be modified or deleted.

IEEE Standards Association
445 Hoes Lane
Piscataway, NJ 08854, USA

This and the following cover pages are not part of the draft. They provide revision and other information for IEEE 802.1 Working Group members and participants in the IEEE Standards Association ballot process, and will be updated as convenient.

The text proper of this draft begins with the Title page.

This draft has been prepared from a set of Framemaker files with conditional text that supports the production of the present amendment draft and a preliminary rollup of that amendment draft into the text of the base standard, IEEE Std 802.1Q-2022 as amended by prior amendments as of the close of their successful SA ballots.

This dated draft contains corrections to cross-references, identified during pre-publication editing of 802.1Qcz.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

P802.1Qdv/D0.4
November 14, 2023
(Amendment to IEEE Std 802.1Q-2022 as amended by IEEE Std 802.1Qcz, P802.1Qcw/D2.2, and P802.1Qcj/D2.6

Draft Standard for Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment: Enhancements to Cyclic Queuing and Forwarding

Prepared by the
Time-Sensitive Networking (TSN) Task Group of IEEE 802.1

Sponsor
LAN/MAN Standards Committee of the IEEE Computer Society

Copyright © 2023 by the IEEE.
Three Park Avenue
New York, New York 10016-5997, USA

All rights reserved.

This document is an unapproved draft of a proposed IEEE Standard. As such, this document is subject to change. USE AT YOUR OWN RISK! IEEE copyright statements SHALL NOT BE REMOVED from draft or approved IEEE standards, or modified in any way. Because this is an unapproved draft, this document must not be utilized for any conformance/compliance purposes. Permission is hereby granted for officers from each IEEE Standards Working Group or Committee to reproduce the draft document developed by that Working Group for purposes of international standardization consideration. IEEE Standards Department must be informed of the submission for consideration prior to any reproduction for international standardization consideration (stds.ipr@ieee.org). Prior to adoption of this document, in whole or in part, by another standards development organization, permission must first be obtained from the IEEE Standards Department (stds.ipr@ieee.org). When requesting permission, IEEE Standards Department will require a copy of the standard development organization's document highlighting the use of IEEE content. Other entities seeking permission to reproduce this document, in whole or in part, must also obtain permission from the IEEE Standards Department.

IEEE Standards Department
445 Hoes Lane
Piscataway, NJ 08854, USA

Abstract: This amendment to IEEE Std 802.1Q-2022 as amended by IEEE Std 802.1Qcz-2023, IEEE Std 802.1Qcw-2023, and IEEE Std 802.1Qcj-2023 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.

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2023 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Unapproved draft.

IEEE and 802 are registered trademarks in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

PDF: ISBN 978-X-XXX-XXX-X STDXXXXX
Print: ISBN 978-X-XXX-XXX-X STDPDXXXXX

IEEE prohibits discrimination, harassment, and bullying. For more information, visit http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (https://standards.ieee.org/ipr/disclaimers.html), appear in all standards and may be found under the heading "Important Notices and Disclaimers Concerning IEEE Standards Documents."

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE Standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE standards documents are supplied "AS IS" and "WITH ALL FAULTS."

Use of an IEEE standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon his or her own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus development process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter's views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group. Statements made by volunteers may not represent the formal position of their employer(s) or affiliation(s).

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents**.

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and subcommittees of the IEEE SA Board of Governors are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the IEEE SA myProject system.[1] An IEEE Account is needed to access the application.

Comments on standards should be submitted using the Contact Us form.[2]

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, neither IEEE nor its licensors waive any rights in copyright to the documents.

[1] Available at: https://development.standards.ieee.org/myproject-web/public/view.html#landing.

[2] Available at: https://standards.ieee.org/content/ieee-standards/en/about/contact/index.html.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; https://www.copyright.com/. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit IEEE Xplore or contact IEEE.[3] For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE SA Website.[4] Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in IEEE Xplore. Users are encouraged to periodically check for errata.

Patents

IEEE Standards are developed in compliance with the IEEE SA Patent Policy.[5]

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at https://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

[3] Available at: https://ieeexplore.ieee.org/browse/standards/collection/ieee.
[4] Available at: https://standards.ieee.org/standard/index.html.
[5] Available at: https://standards.ieee.org/about/sasb/patcom/materials.html.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

IMPORTANT NOTICE

IEEE Standards do not guarantee or ensure safety, security, health, or environmental protection, or ensure against interference with or from other devices or networks. IEEE Standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Participants

<<The following lists will be updated in the usual way prior to publication>>

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the IEEE 802.1 Working Group had the following membership:

Glenn Parsons, *Chair*
Jessy V. Rouyer, *Vice Chair*
János Farkas, *Chair, Time-Sensitive Networking Task Group*
Craig Gunther, *Vice Chair, Time-Sensitive Networking Task Group*
Norman Finn, *Editor*

<<TBA>>

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

<<TBA>>

When the IEEE-SA Standards Board approved this standard on XX Month 20xx, it had the following membership:

<<TBA>>

<<TBA>>

*Member Emeritus

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Introduction

This introduction is not part of IEEE Std 802.1Qdv™-2024, IEEE Standard for Local and metropolitan area networks—Bridges and Bridged Networks—Amendment 38: Enhancements to Cyclic Queuing and Forwarding.

IEEE Std 802.1Q™dv-2024: Enhancements to Cyclic Queuing and Forwarding 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.

This standard contains state-of-the-art material. The area covered by this standard is undergoing evolution. Revisions are anticipated within the next few years to clarify existing material, to correct possible errors, and to incorporate new related material. Information on the current revision state of this and other IEEE 802 standards may be obtained from

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854-4141
USA

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Contents

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Figures

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Tables

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

IEEE Standard for Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment: Enhancements to Cyclic Queuing and Forwarding

(This amendment is based on IEEE Std 802.1Q™-2022 as amended by IEEE Std 802.1Qcz™-2023, IEEE Std 802.1Qcw™-2023, and IEEE Std 802.1Qcj™-2023.)

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]

1. Overview

1.3 Introduction

Insert the following text at the end of 1.3:

This standard specifies procedures, protocols and managed objects for Bin Cyclic Queuing and Forwarding (BCQF). To this end, it

dg) 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;

[1] Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

dh) Specifies a procedure for storing received frames into bins based on the time of reception of the frame.

di) Specifies a procedure for storing received frames into bins based on per-flow octet counters.

dj) Specifies a protocol for determining the phase relationship between a transmitter's and a receiver's bin boundaries in time.

dk) Provides managed objects, Management Information Base (MIB), and YANG modules for controlling these procedures.

dl) Provides an informative annex to provide guidance for applying these procedures.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

2. Normative references

Insert the following items into the list of Normative References:

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

3. Definitions

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

<< Editor's note: No new terms have so far been selected for inclusion, Note that "Cyclic Queuing and Forwarding" is not defined in IEEE Std 802.1Q-2022. >>

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

4. Abbreviations

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

BCQF	Bin Cyclic Queuing and Forwarding
cCQF	count-based Cyclic Queuing and Forwarding
CPAP	Cyclic queuing and forwarding (CQF) Phase Alignment Protocol
sCQF	scheduled Cyclic Queuing and Forwarding
tCQF	time-based Cyclic Queuing and Forwarding

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

5. Conformance

5.4 VLAN Bridge component requirements

5.4.1 VLAN Bridge component options

Change 5.4.1.9 as follows:

5.4.1.9 Cyclic Queuing and Forwarding (CQF) requirements

5.4.1.9.1 Scheduled Cyclic queuing and forwarding (sCQF) requirements

A VLAN Bridge component implementation that conforms to the provisions of this standard for ~~CQF~~sCQF (see Annex T) shall:

a) Support the enhancements for scheduled traffic as specified in 8.6.8.4.
b) Support the state machines for scheduled traffic as specified in 8.6.9.
c) Support the state machines for stream gate control as specified in 8.6.10.
d) Support the management entities for scheduled traffic as specified in 12.29.
e) Support the requirements for Per-Stream Filtering and Policing (PSFP) as stated in 5.4.1.8.
f) Support the management entities for PSFP as specified in 12.31.

5.4.1.9.2 Bin Cyclic Queuing and Forwarding (BCQF) requirements

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

a) Support the BCQF transmission selection algorithm as specified in 8.6.8.7.
b) Support the management entities for BCQF as specified in 100.1.2, 100.1.3, and 100.1.4.

5.4.1.9.3 Time-based Cyclic Queuing and Forwarding (tCQF) requirements

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

a) Support BCQF as stated in 5.4.1.9.2.
b) Support time-based CQF bin selection as specified in 8.6.5.4.
c) Support the management entities for time-based CQF as specified in 100.1.1.

5.4.1.9.4 Count-based Cyclic Queuing and Forwarding (cCQF) requirements

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

a) Support BCQF as stated in 5.4.1.9.2.
b) Support count-based CQF bin selection as specified in 8.6.5.5.
c) Support the management entities for count-based CQF as specified in 100.1.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

5.13 MAC Bridge component requirements

5.13.1 MAC Bridge component options

Change 5.13.1.2 as follows:

5.13.1.2 Cyclic Queuing and Forwarding (CQF) requirements

5.13.1.2.1 Scheduled Cyclic queuing and forwarding (sCQF) requirements

A MAC Bridge component implementation that conforms to the provisions of this standard for Scheduled CQF (see Annex T) shall:

a) Support the enhancements for scheduled traffic as specified in 8.6.8.4.
b) Support the state machines for scheduled traffic as specified in 8.6.9.
c) Support the state machines for stream gate control as specified in 8.6.10.
d) Support the management entities for scheduled traffic as specified in 12.29.
e) Support the requirements for PSFP as stated in 5.13.1.1.
f) Support the management entities for PSFP as specified in 12.31.

5.13.1.2.2 Bin Cyclic Queuing and Forwarding (BCQF) requirements

A MAC Bridge component implementation that conforms to the provisions of this standard for Bin CQF (see Annex Y) shall

a) Support the BCQF transmission selection algorithm as specified in 8.6.8.7.
b) Support the management entities for BCQF as specified in 100.1.2, 100.1.3, and 100.1.4.

5.13.1.2.3 Time-based Queuing and Forwarding (tCQF) requirements

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

a) Support BCQF as stated in 5.13.1.2.2.
b) Support time-based CQF bin selection as specified in 8.6.5.4.
c) Support the management entities for time-based CQF as specified in 100.1.1.

5.13.1.2.4 Count-based Queuing and Forwarding (cCQF) requirements

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

a) Support BCQF as stated in 5.4.1.9.2.
b) Support count-based CQF bin selection as specified in 8.6.5.5.
c) Support the management entities for count-based CQF as specified in 100.1.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Change 6.28 as follows:

5.28 End station requirements—Cyclic queuing and forwarding (CQF)

5.28.1 End station requirements—Scheduled Cyclic queuing and forwarding (sCQF)

An end station implementation that conforms to the provisions of this standard for Scheduled CQF (see Annex T) shall:

a) Support the enhancements for scheduled traffic as specified in 8.6.8.4.
b) Support the state machines for scheduled traffic as specified in .
c) Support the state machines for stream gate control as specified in 8.6.10.
d) Support the management entities for scheduled traffic as specified in 12.29.
e) Support the requirements for PSFP as stated in 5.27.
f) Support the management entities for PSFP as specified in 12.31.

5.28.2 End station requirements for count-based CQF

An end station implementation that conforms to the provisions of this standard for count-based Cyclic Queuing and Forwarding (cCQF) (see Annex Y) shall

a) Support the BCQF transmission selection algorithm as specified in 8.6.8.7.
b) Support the management entities for BCQF as specified in 100.1.3..
c) Support count-based CQF bin selection as specified in 8.6.5.5.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

6. Support of the MAC Service

6.5 Quality of service (QoS) maintenance

6.5.2 Frame loss

Change 6.5.2 as follows:

The MAC Service does not guarantee the delivery of Service Data Units (SDUs). Frames transmitted by a source station arrive, uncorrupted, at the destination station with high probability. The operation of a Bridge introduces minimal additional frame loss.

A frame transmitted by a source station can fail to reach its destination station as a result of:

a) Frame corruption during physical layer transmission or reception.
b) Frame discard by a Bridge because:
   1) It is unable to transmit the frame within some maximum period of time and, hence, must discard the frame to prevent the maximum frame lifetime (6.5.6) from being exceeded.
   2) It is unable to continue to store the frame due to exhaustion of internal buffering capacity as frames continue to arrive at a rate in excess of that at which they can be transmitted.
   3) The size of the SDU carried by the frame exceeds the maximum supported by the MAC procedures employed on the LAN to which the frame is to be relayed.
   4) Changes in the connected topology of the network necessitate frame discard for a limited period of time to maintain other aspects of QoS (13.16).
   5) The device attached to the Bridge is not authorized for access to the network.
   6) The configuration of Static Filtering Entries or Static VLAN Registration Entries in the FDB (8.8.1, 8.8.2) disallows the forwarding of frames with particular destination addresses or VLAN classifications on specific Ports.
   7) A flow metering algorithm (8.6.5) determines that discard is necessary.
   8) The Bridge supports enhancements for scheduled traffic (8.6.8.4) and the size of the service data unit exceeds the value of queueMaxSDU (8.6.8.4) for the traffic class queue on which the frame is to be queued.
   9) The Bridge supports Bin Cyclic Queuing and Forwarding (BCQF, (8.6.8.7) and the frames assigned to a bin in the class of service queue cannot be transmitted before the end of the transmission period alloted that bin.

NOTE—As Static Filtering Entries and Static VLAN Registration Entries are associated with particular Ports or combinations of Ports, there is a possibility that misconfiguration of such entries will lead to unintended frame discard during or following automatic reconfiguration of the network.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

8. Principles of Bridge operation

8.6 The Forwarding Process

8.6.5 Flow classification and metering

Change 8.6.5 as follows:

The Forwarding Process can apply flow classification and metering to frames that are received on a Bridge Port and have one or more potential transmission ports. Bridge Ports and end stations may support Per-Stream Filtering and Policing (PSFP), Asynchronous Traffic Shaping (ATS) filtering and eligibility time assignment, Bin Cyclic Queuing and Forwarding (BCQF) and eligibility time (egress bin) assignment, Congestion Isolation (CI), or the general flow classification rules specified in 8.6.5.1.

NOTE—The general flow classification and metering specification was added to this standard by IEEE Std 802.1Q-2005, PSFP by IEEE Std 802.1Qci-2017, ATS by IEEE Std 802.1Qcr-2020, BCQF by IEEE Std 802.1Qdv-2024, and CI by IEEE Std 802.1Qcz-2023.

PSFP, ATS, and CI share common per-stream classification and metering elements, as shown in Figure 8-13. The Stream identification function specified in IEEE Std 802.1CB can be used to associate received frames with these elements. BCQF can use either general or per-stream classificationand metering.

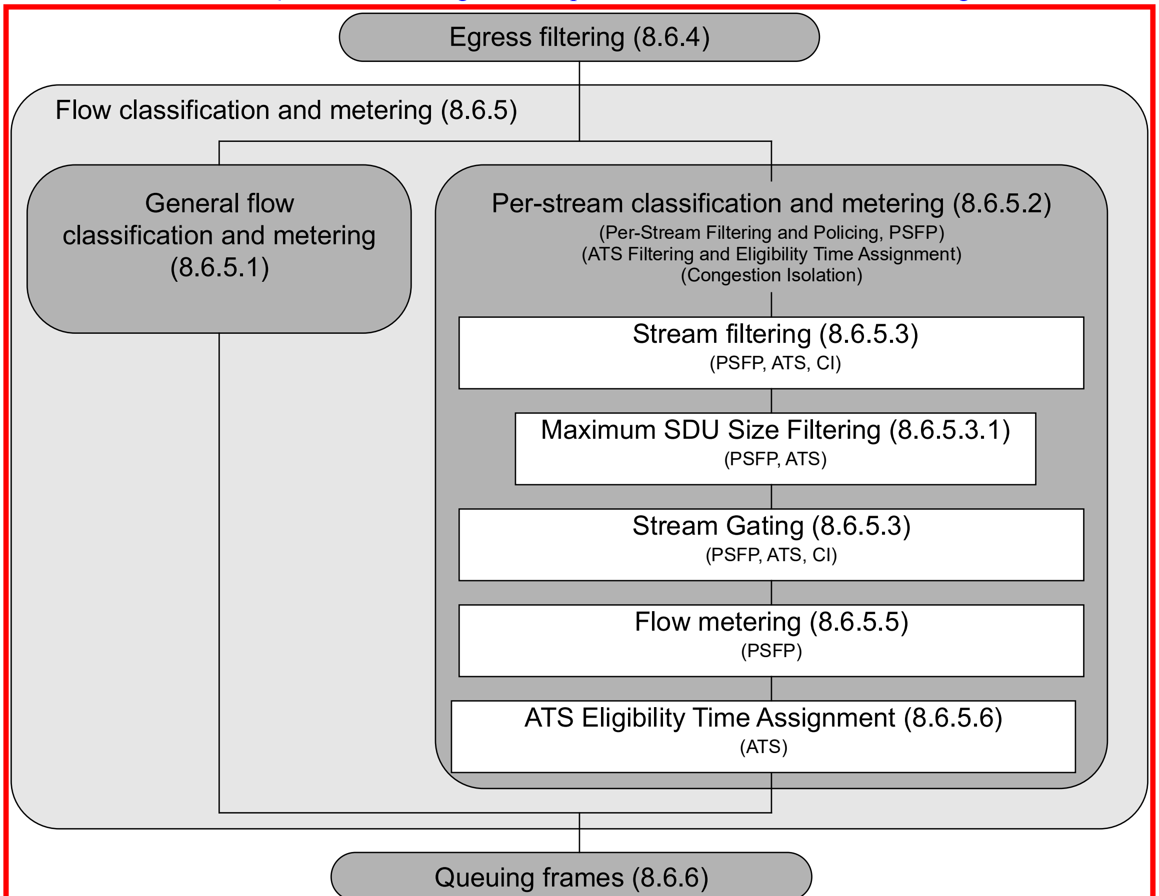


Figure 8-13—Flow classification and metering

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

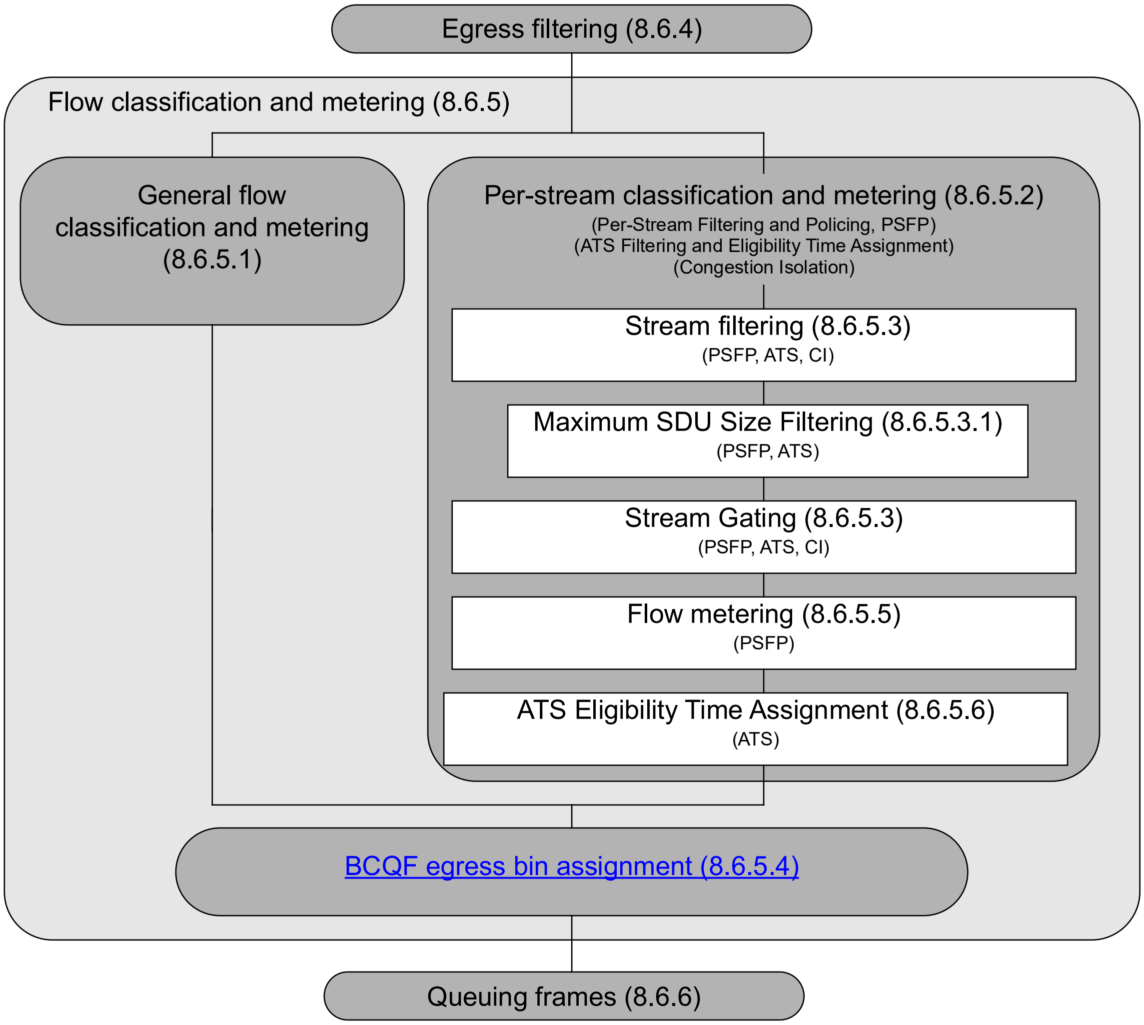


Figure 8-13—Flow classification and metering

Change 8.6.5.2 as follows:

8.6.5.2 Per-stream classification and metering

When Per-Stream Filtering and Policing (PSFP), Asynchronous Traffic Shaping (ATS), or Congestion Isolation (CI) is used, filtering and policing decisions for received frames are made, and subsequent BCQF bin assignment (8.6.5.4), queuing (8.6.6) and transmission selection decisions (8.6.8) supported, as follows:

a) Each received frame can be associated with a stream filter, as specified in 8.6.5.3. If a matching stream filter is specified (8.6.5.3), that is used to process the frame. Wildcard stream filters can be configured to match and discard frames not associated with a specified stream. If no matching stream filter is found, the frame is passed on for BCQF bin assignment (8.6.5.4) and ~~queued for~~ transmission ~~as specified in~~ (8.6.6).

b) If the stream filter specifies maximum SDU size filtering (8.6.5.3.1), that is used to process the frame. The frame can be discarded if a maximum SDU size is exceed. The ATS scheduler state machine operation (8.6.11) assumes that the sizes of frames that it processes are less than or equal to the associated CommittedBurstSize parameter (8.6.11.3.5).

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

c) The stream filter specifies a stream gate (8.6.5.3), that is used to process the frame. The frame can be discarded if it is received outside of permitted reception intervals or a given data limit within a reception interval is exceeded. The frame's priority can be mapped to an internal priority value (IPV) that can influence subsequent queuing decisions (8.6.5.4, 8.6.6).

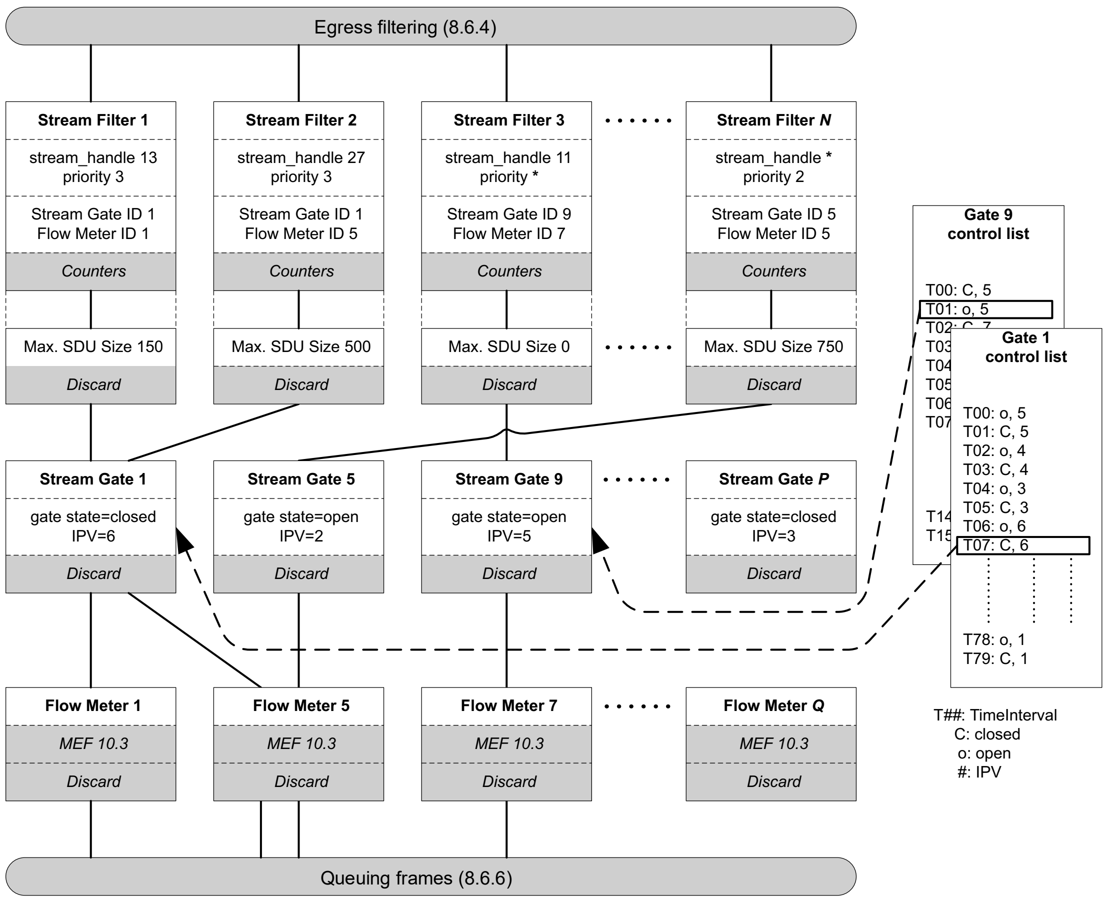
d) If the stream filter specifies a flow meter (8.6.5.5), that is used to process the frame. The frame can be discarded or marked as drop eligible if a traffic limit of a flow meter is exceeded. A given stream filter can be configured with flow meters and an ATS scheduler if both PSFP and ATS are supported.

e) If the stream filter specifies an ATS scheduler (8.6.5.6), that is used to process the frame. It computes an eligibility time for the frame for subsequent use by the ATS transmission selection algorithm (8.6.8.5). The frame can be discarded if a maximum eligibility time is exceeded (8.6.11.3.13).

8.6.5.2.1 PSFP support

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Change Figure 8-14 as follows:

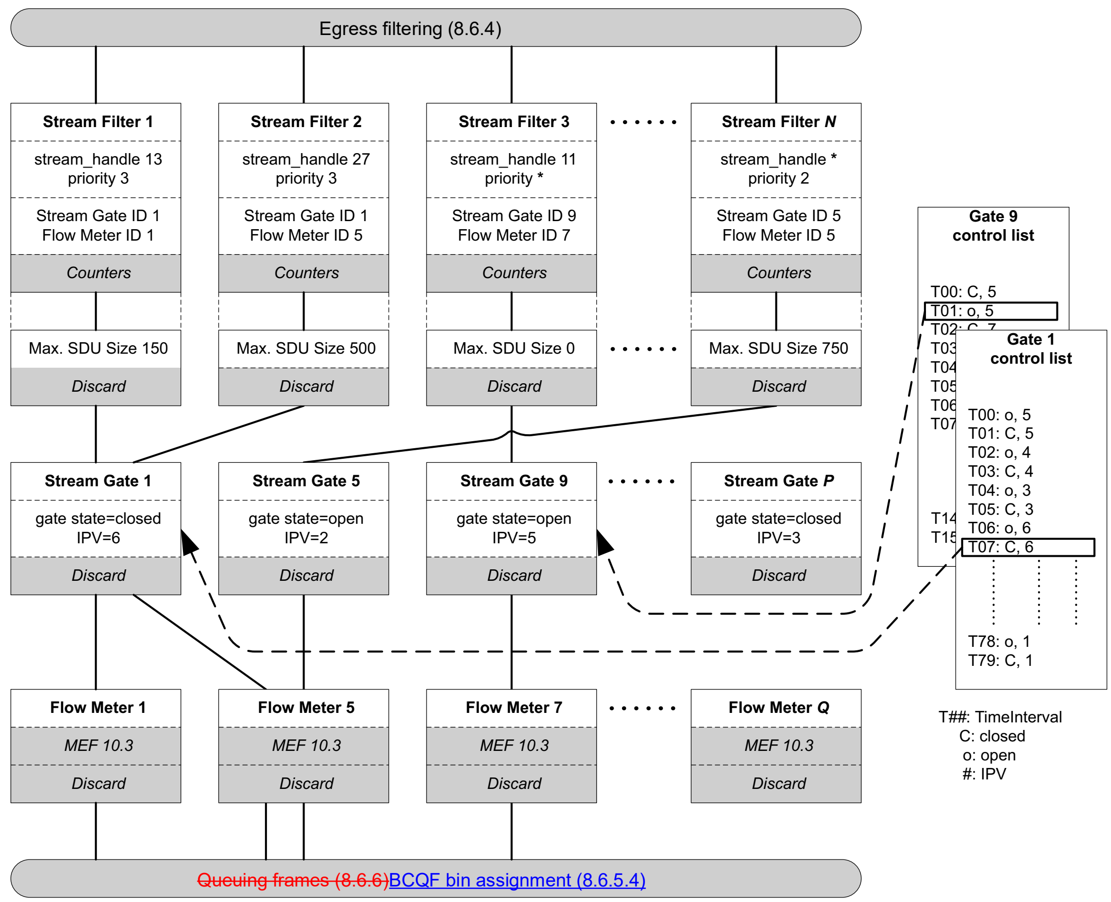


KEY
Counters: Matching, passing, and discarded frame counters (8.6.5.3).
Discard: Frame discarding abilities and parameters (8.6.5.3.1, 8.6.5.3, 8.6.5.5).
MEF 10.3: Flow metering based on MEF 10.3 Bandwidth Profile parameters and algorithm as specified in 8.6.5.5.

Figure 8-14—Per-stream classification for PSFP

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

8.6.5.2.2 ATS support

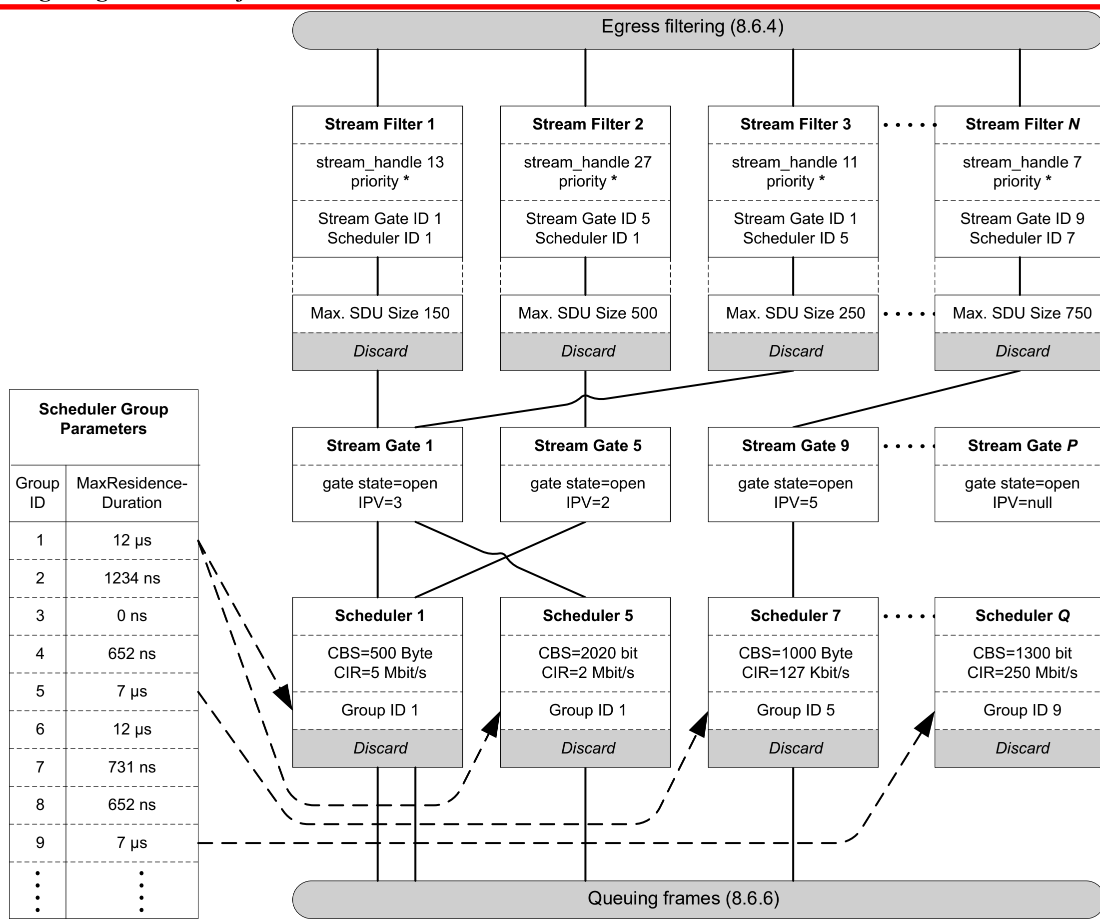


KEY
Counters: Matching, passing, and discarded frame counters (8.6.5.3).
Discard: Frame discarding abilities and parameters (8.6.5.3.1, 8.6.5.3, 8.6.5.5).
MEF 10.3: Flow metering based on MEF 10.3 Bandwidth Profile parameters and algorithm as specified in 8.6.5.5.

Figure 8-14—Per-stream classification for PSFP

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

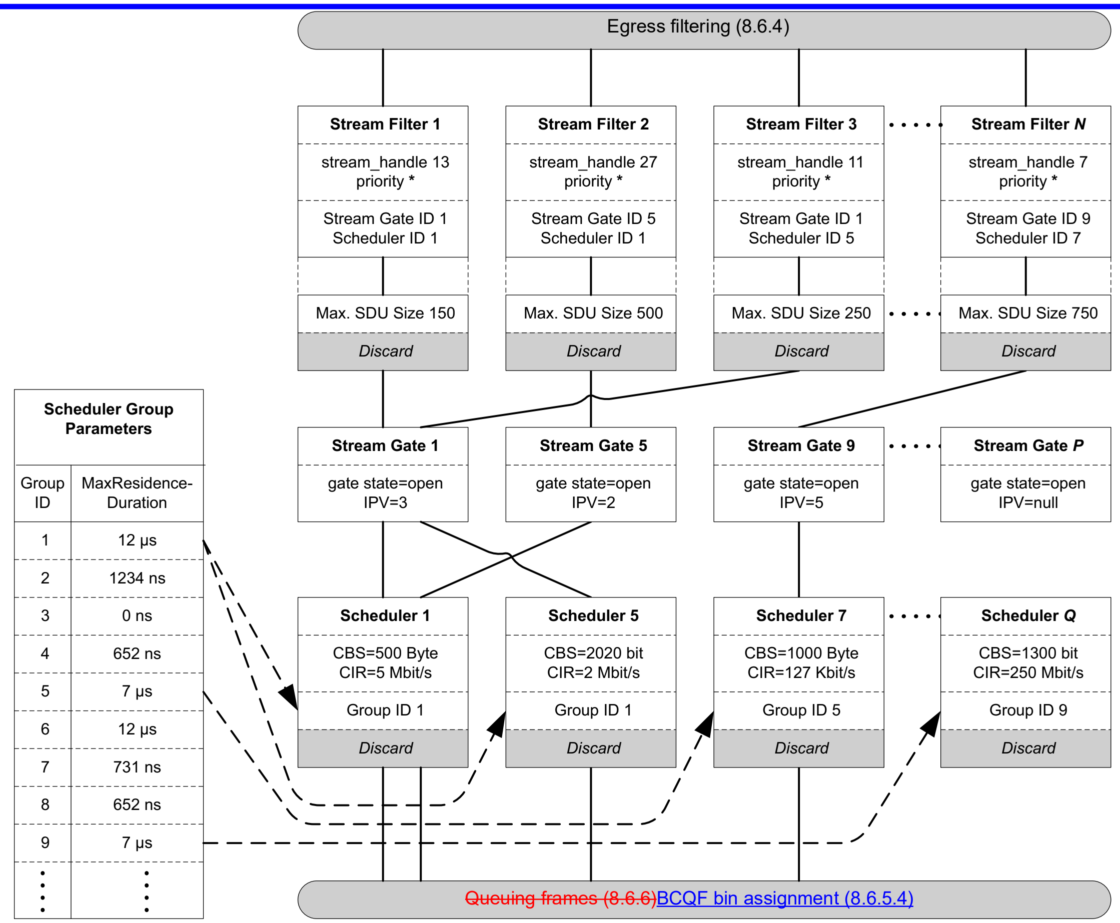
1 ***Change Figure 8-15 as follows:***



KEY
Discard: Frame discarding abilities and parameters (Clause 8.6.5.3.1, Clause 8.6.5.3, 8.6.5.6).
CBS: CommittedBurstSize parameter (8.6.5.6, 8.6.11.3.5).
CIR: CommittedInformationRate parameter (8.6.5.6, 8.6.11.3.6).

Figure 8-15—Per-stream classification and metering for ATS

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.



KEY
Discard: Frame discarding abilities and parameters (Clause 8.6.5.3.1, Clause 8.6.5.3, 8.6.5.6).
CBS: CommittedBurstSize parameter (8.6.5.6, 8.6.11.3.5).
CIR: CommittedInformationRate parameter (8.6.5.6, 8.6.11.3.6).

Figure 8-15—Per-stream classification and metering for ATS

Change the first paragraph of 8.6.5.4 as follows:

8.6.5.3 Stream gating

Stream gates can discard frames whose reception times contradict a given time schedule. Stream gates can also map the frame's priority to an internal priority value (IPV) that is used to make subsequent queuing decisions (8.6.5.4, 8.6.6), while retaining the frame's original priority for transmission.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Insert the following at the end of 8.6.5:

8.6.5.4 tCQF time-based bin assignment

Time-based CQF (tCQf) assigns an *bin_number* to a frame in two steps:

a) The frame is assigned an *ingress_cycle* based on its ingress port and priority/IPV;, as controlled by the TcqfingressTable (100.1.1). See 8.6.5.4.1.

b) The *ingress_cycle* is mapped to a *bin_number* based on the ingress port, egress port, stream_handle, and priority/IPV, as controlled by the BcqfStreamControlTable (100.1.2). and the BcqfEgressTable (100.1.3). See 8.6.5.4.2.

An ingress cycle assignment state machine can be configured by means of the TcqfIpvTable (100.1.1.4). The state machine is tied to an ingress port and a value of priority/IPV. If a frame's ingress port and the value of its priority (IPV, if assigned by stream gating, 8.6.5.3) match that of an ingress cycle assignment state machine (configured by means of the (TcqfIpvTable, 100.1.1.4), that state machine assigns to the frame an integer *ingress_cycle*. The *ingress_cycle* value can then be used by the egress bin assignment function (8.6.5.4.2) to assign the frame a *bin_number* to be used by the BCQF transmission selection algorithm (8.6.8.7).

8.6.5.4.1 tCQF ingress cycle assignment

There is one ingress cycle assignment state machine per port on which BCQF frames can be received, per traffic class configured for BCQF. Each state machine is governed by an entry in an TcqfIpvTable (100.1.1.4), which itself is an entry in the TcqfingressTable (100.1.1). A received frame is process by state machine whose IPV and ingress port number matches those of the frame.

The ingress cycle assignment state machine assigns an *ingress_cycle* to the frame. This is an integer value that is in the range 0 through *n_max* – 1, where *n_max* is greater than or equal to the largest number of bins allocable to any BCQF egress port.

The value for *ingress_cycle* is derived from the time that the first bit of the frame was received, *ingress_time_stamp*. Clause 99 of IEEE Std 802.3 can be used to supply this value. The following entries in the TcqfingressTable (100.1.1) are also used:

a) TcqfingressEpoch,(100.1.1.2) from the TcqfingressTable (100.1.1).

b) TcqfingressPeriod,(100.1.1.4.3) from the TcqfIpvTable (100.1.1.4).

Then,

$$ingress\_cycle = ((ingress\_time\_stamp - \text{TcqfingressEpoch}) / \text{TcqfingressPeriod}) \text{ modulo } n\_max$$

8.6.5.4.2 tCQF egress bin assignment

Time-based CQF applies only to frames matching an entry in the BcqfStreamControlTable (100.1.2) that has BcqfBinSelectionMode (100.1.2.4.5) set to **time_based_assignment**.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

When a frame has been assigned to a specific class of service queue that is configured for BCQF (8.6.8, Table), then the tCQF egress bin assignment function computes a separate *bin_number* for the frame. This *bin_number* is then used by the BCQF transmission selection algorithm (8.6.8.7) to enable the frame for transmission selection. Each translation is controlled by:

a) The *ingress_cycle* value assigned by tCQF ingress cycle assignment (8.6.5.4.1).
b) The values of BcqfIntentionalDelayBins (100.1.2.4.4) and BcqfMaximumExtraCcqfBins (100.1.2.4.6). These values are from the BcqfStreamControlTable (100.1.2), indexed by the ingress port and egress port, or by stream_handle and egress port, of the frame.
c) The current value *n_bins* and the current value of *transmitting_bin* in the BCQF transmission state machine (8.6.8.7).
d) An integer value *bin_offset* from the *cycle_phase_offset_table*. This is notionally a table of integers, indexed by ingress port, egress port, and class of service, that adjusts the *ingress_cycle* to an offset from the egress port's *transmitting_bin.*

Time-based CQF (tCQF) assigns a *bin_number* in the range 0 through *n_bins* to the frame as follows:

bin_number = *(ingress_cycle* + *transmitting_bin* + *bin_offset* + BcqfIntentionalDelayBins) modulo *n_bins*

NOTE 1—Computing values for the *cycle_phase_offset_table* s not trivial. See Clause Y in general, and Y.3.11.3 in particular, for an explanation of factors that influence this computation.

For every entry in the BcqfStreamControlTable (100.1.2) that matches a particular egress port and class of service, and has the BcqfBinSelectionMode (100.1.2.4.5) set to **time_count_assignment**, a state machine is instantiated. That state machine tracks the total number of transmit bit times for the frames matching the entry that have been deposited in the bin to which such frames are being assigned.. If the number of transmit bit time deposited in the bin would exceed the value in CcqfAllocatedBits, that frame is discarded.

<< Editor's note: Marking down the priority, instead of discarding, is also a possibility. This would require another entry in the BcqfStreamControlTable, specifying the markdown priority. Would this be worth the cost of specifying? If implementing? Does either one really accomplish what we want? Ballot comments solicited. >>

NOTE 2—The above description leaves it unclear as to whether frames can be deposited in a bin that is transmitting. If limitations are placed on fan-in (BcqfMaximumFanIn, 100.1.3.6.5) it is possible to do so in certain circumstances. See also the notes in 100.1.2.4.2 and 100.1.2.4.3.

8.6.5.5 cCQF count-based bin assignment

Count-based CQF (cCQF) applies only to frames matching an entry in the BcqfStreamControlTable (100.1.2) that has BcqfBinSelectionMode (100.1.2.4.5) set to **count_based_assignment** or **time_count_assignment**.

cCQF assigns a *bin_number* to each frame by operating a state machine for each stream, egress port, and each class of service queue that is configured in the BcqfStreamControlTable 100.1.2) with the value TRUE in the BcqfStreamControlTable (100.1.2) object. When a frame matching that entry is assigned to that queue, the *bin_number* is assigned subject to the following constraints:

a) Every egress bin in the queue is in one of four states: transmitting, full, filling, or empty.
b) The bin identified by the transmission selection algorithm (8.6.8.7) as the *transmitting_bin* is in the transmitting state.
c) When the state machine is initialized, the next-higher-numbered bin after the transmitting bin, monulo *n_bins* (8.6.8.7) is in the filling state; all other bins are in the empty state.
d) When the transmission selection algorithm (8.6.8.7) advances the *transmitting_bin*, the first bin in the filling (or full) state becomes the transmitting bin, the bin succeeding the old transmitting bin becomes filling, if it was empty, and the old transmitting bin goes to the in the empty state"

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

e) The state machine maintains a count of transmission bit times deposited into the filling bin for this stream.

f) If depositing a frame into the filling bin would not cause the total number of transmission time bits to exceed the limit specified by CcqfAllocatedBits (100.1.2.4.1), the frame is assigned the *bin_number* of the filling bin, and the total is updated. Otherwise, the bin is placed in the full state, and the succeeding (empty) bin is placed in the filling state, and its *bin_number* is assigned to the frame.

g) If step f) would exceed the number of extra bins allocable by BcqfMaximumExtraCcqfBins (100.1.2.4.6), the frame id discarded.

h) The ordering requirements specified in 8.6.6 are met.

NOTE 1—This formulation is presented from a slightly different perspective than in the Seaman paper [B2]. This standard describes four **bin states**, transmitting full, filling, and empty; [B2] describes four **queue names**, *prior*, *current*, *next*, and *last*.Tthe bin in the transmitting state corresponds exactly to the *prior* queue.. The correspondence is less exact for the other terms.

NOTE 2—This formulation assumes that BcqfMaximumExtraCcqfBins (100.1.2.4.6) ensures that, when advancing from a full bin to the next, one never wraps around to the transmitting bin.

NOTE 3—The above description does not allow frames to be deposited in a bin that is transmitting. If limitations are placed on fan-in (BcqfMaximumFanIn, 100.1.3.6.5) it is possible to do so in certain circumstances. See also the notes in 100.1.2.4.2 and 100.1.2.4.3.

8.6.8 Transmission selection

Insert the following at the end of 8.6.8:

8.6.8.7 BCQF transmission selection algorithm

Every frame placed into a class of service queue configured for BCQF is assigned a *bin_number* (8.6.5.4, 8.6.5.5). Each BCQF supports *n_bins* values for *bin_number*, 0 through *n_bins*–1. A BCQF queues has a variable *transmitting_bin*, which determines which bin number is eligible for transmission selection. Only frames whose *bin_number* matches *transmitting_bin* are eligible for selection for transmission. The value of *transmitting_bin* is periodically incremented, modulo *n_bins*, thus rotating the transmitting bin through all *n* bins. Bin rotation occurs at intervals of *BCQF_cycle* seconds, starting at the rime *BCQF_cycle_start*. The time for bin rotation is determined by a BCQF Cycle Clock, which is an implementation specific local system clock function.

a) *n_bins*: The number of bins configured for this BCQF queue.

b) *transmitting_bin*: The number of the bin currently eligible for selection for transmitting frames.

c) *BCQF_cycle_start*: The starting time from which bin rotation occurs every *BCQF_cycle* seconds.

d) *BCQF_cycle*: A rational time in seconds between bin rotation events.

If, a bin cannot be completely drained of all frames before it is rotated out and the succeeding bin enabled for transmission selection, the remaining frames shall be discarded. That is, whenever the value of *transmitting_bin* is incremented, all frames with that value for *bin_number* remaining in the queue are discarded. No frame can be selected for transmission unless its transmission can be completed before the *transmitting_bin* rotates.

NOTE—The “notional” division of a queue into bins implies no specific implementation. The ATS transmission selection algorithm (8.6.8.5) could be used, for example, by assigning the same eligibility time to the frames in the same notional bin.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

46. Time-Sensitive Networking (TSN) configuration

46.1 Overview of TSN configuration

46.1.3 TSN configuration models

46.1.3.2 Centralized network/distributed user model

Change the bulleted list in 46.1.3.2 as follows:

The following TSN features can be configured by the CNC using this model:

a) Credit-based shaper algorithm (8.6.8.2) and its configuration (Clause 34)
b) Frame preemption (6.7.2)
c) Scheduled traffic (8.6.8.4, 8.6.9)
d) Frame Replication and Elimination for Reliability (IEEE Std 802.1CB)
e) Per-Stream Filtering and Policing (8.6.5.1)
f) Cyclic queuing and forwarding (Annex T, Annex Y)

46.1.3.3 Fully centralized model

Change the bulleted list in 46.1.3.3 as follows:

The following TSN features can be configured by the CNC using this model:

a) Credit-based shaper algorithm (8.6.8.2) and its configuration (Clause 34)
b) Frame preemption (6.7.2)
c) Scheduled traffic (8.6.8.4, 8.6.9)
d) Frame Replication and Elimination for Reliability (IEEE Std 802.1CB)
e) Per-Stream Filtering and Policing (8.6.5.1)
f) Cyclic queuing and forwarding (Annex T, Annex Y)

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Insert the following Clause:

98. Generic Dot1Q Protocol

98.1 Overview of GD1QP

In order to conserve EtherTypes, a format for a Generic Dot1Q Protocol (GD1QP) is specified in this clause. GD1QP supports up to 256 GD1Q User protocols, all employing frames encoded with the same EtherType. GD1QP sends and receives the Generic Dot1Q Protocol Data Units (GD1QPDUs) described in 98.2, and conforms to the operational requirements specified in 98.3.

98.2 Frame format

The format of a QD1QP PDU are shown in Table 98-1. The fields are defined in the following clauses

Table 98-1—Generic Dot1Q Data Unit Format

	Octet	Length
EtherType (98.2.1)	0	2
ProtocolId (98.2.2)	2	1
Version (98.2.3)	3	1
Protocol dependent {98.2.4)	4	

98.2.1 EtherType

The EtherType is of the Generic Dot1Q Protocol is specified in Table 98-2.

Table 98-2—Generic Dot1Q EtherType

EtherType	Value
Generic Dot1Q Protocol	0xYYYY[a]

[a]To be determined before Standards Association Ballot

98.2.2 ProtocolId

The Protocol ID field is a value taken from Table 98-3.

98.2.3 Version

The use of the version field is defined by the specification of a particular GD1Q User protocol. The expected use is to allow alterations to the format of the protocol dependent fields while maintaining backwards compatibility (new implementations can understand old versions) and forwards compatibility (old implementations can identify what parts of a new version contain understandable information and what parts are new, and can be ignored.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Table 98-3—Generic Dot1Q Protocol ID values

Protocol	ProtoclId
CQF Phase Alignment Protocol (Clause 99)	0
reserved for future use by IEEE 802.1	1-255

98.2.4 Protocol dependent

The use of the rest of the frame is defined by the specification of a particular GD1Q User protocol.

98.3 GD1Q User protocol requirements

A system that receives a frame with a GD1QPDU with a value in the ProtocolId field (98.2.2) does not correspond to a protocol in Table 98-3 that is implemented in the system shall discard the frame.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Insert the following Clause:

99. CQF Phase Alignment Protocol

99.1 Overview of CPAP

See Annex Y for an explanation of Bin Cyclic Queuing and Forwarding (BCQF), to which the CQF Phase Alignment Protocol (CPAP) applies.

Figure 99-1 is an abridgment of Figure Y-4. It shows an example of BCQF. Bridge A is transmitting from a BCQF queue to Bridge B. Whether it is using time-based CQF (Y.3), count-based CQF (Y.4) or some other method to fill its bins is irrelevant. Bridge B is using time-based CQF to assign frames to its egress bins (not shown). Timeline 1 is the egress BCQF timeline in the transmitting Bridge. Timeline 2 is the receiving time-base CQF timeline in the receiving Bridge. The BCQF transmitting timeline for the receiving Bridge (see Y.3.2) is not shown in Figure 99-1. The time ticks on each timeline in Figure 99-1 indicate the start/end of a cycle of duration T_C. These ticks are defined in terms of the transmit timestamp values of transmitted frames (IEEE Std 802.3 clause 90).

\As described in Y.3.2, Bridge B needs to assign each received Stream frame to an egress queue bin, based solely on the time of arrival of the frame at Bridge B's ingress port (IEEE Std 802.3 clause 90). In order to assign frames to bin based only on time, Bridge B runs its egress cycles with exactly the same period T_C as Bridge A (see Y.3.1), though not necessarily in phase (synchronized). The problem to be solved by CPAP is described in Y.3.2.2. Bridge B needs to establish the frame arrival time at its ingress port that corresponds to a transmit cycle boundary in Bridge A. That is, Bridge B wants to know what ingress timestamp it would expect to see on a frame Bridge A transmitted at exactly the start of a cycle in Bridge A.

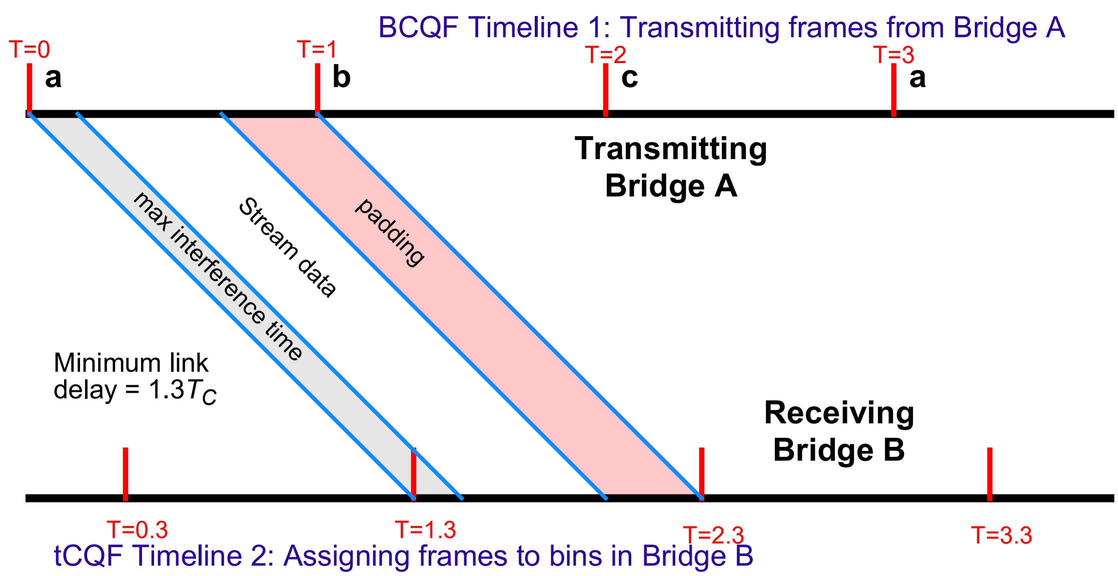


Figure 99-1—Aligning the transmitter and receiver BCQF cycle start times

The transmit timestamp in Bridge A is a local matter; its format and frequency are not visible outside Bridge A, and the timestamp is not a part of the transmitted frame. Similarly, the ingress timestamp in Bridge B is local to that Bridge and bears no relationship to the transmit timestamp in Bridge A, except that, when translated to seconds of elapsed time, both Bridges' timestamps advance at a rate close to their respective BCQF cycles. (Such variations are included in the definition of T_V in Y.3.2.)

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

99.2 CPAP procedures

Figure 99-2 illustrates the operation of CPAP. Two systems are involved, a CPAP transmitter and a CPAP receiver. The CPAP transmitter is presumed to also be transmitting, or preparing to transmit, data frames from one or more BCQF-enabled queues. The transmission of two CPAP messages are required, both transmitted from the CPAP transmitter: a CPAP Time Marker Frame, and a CPAP Phase Offset Message frame. See 99.4 for the formats of these messages.

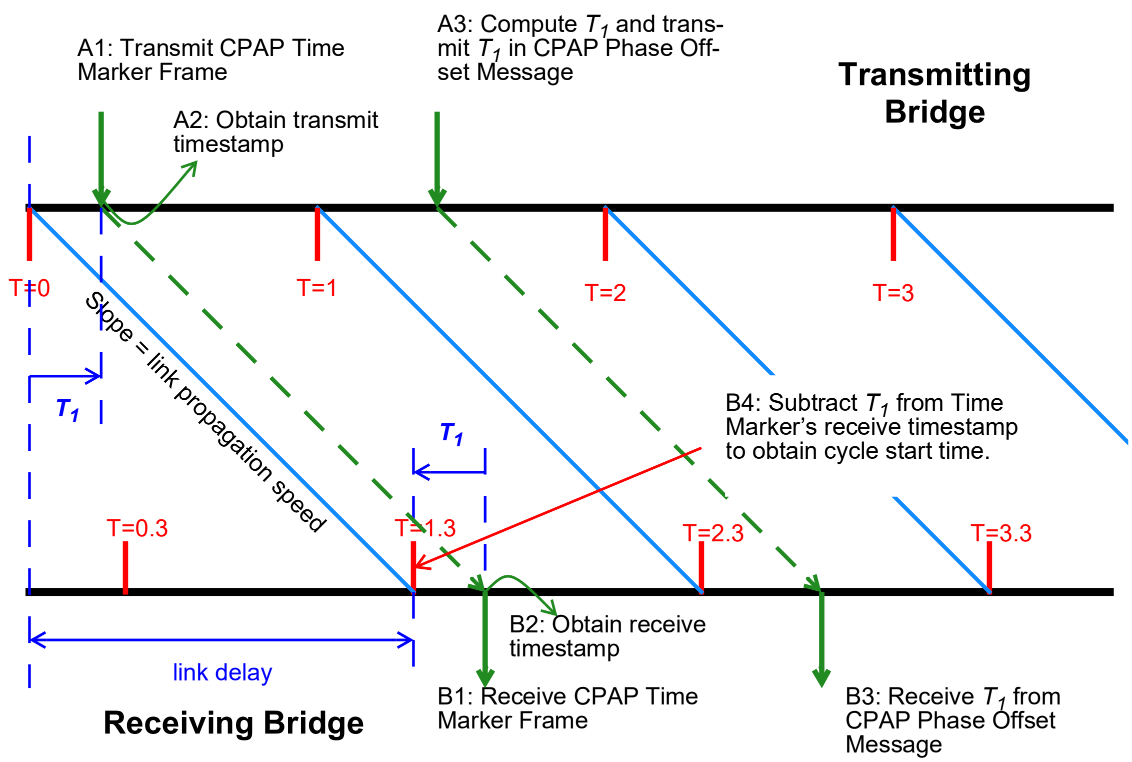


Figure 99-2—CQF Phase Alignment Protocol sequence

There is no implied relationship among the relative timing of cycle start times and CPAP messages beyond the requirements in 99.3. The two message need not be transmitted either in the same or in different cycles. It is a system administrator's choice as to what priority CPAP message are sent, and whether a Stream reservation is established for them. The default is priority 0, which is presumed to be a best-effort priority.

One CPAP sequence consists of the transmission of a CPAP Time Marker Frame followed by the transmission of a CPAP Phase Offset Message. The combination of the time between the two messages of a CPAP sequence, and the frequency of the CPAP sequences, contribute to the accuracy of the resultant. See 99.4.

If the BCQF transmitter is operating more than one class of service queue with BCQF enabled, the CPAP sequence is applied only to the slowest cycle (largest value of T_C).

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

The sequence of events in the CPAP transmitter, for one CPAP sequence, is as follows:

a) (Event A1 in Figure 99-2.) The CPAP transmitter transmits a CPAP Time Marker Frame.
b) (Event A2 in Figure 99-2.) The CPAP transmitter obtains a value, in locally-meaningful units, for the time at which the first bit of the CPAP Time Marker Frame was placed on the medium connecting the two CPAP systems (see IEEE Std 802.3 Clause 90).
c) (Event A3 in Figure 99-2.) The CPAP transmitter calculates time interval T_I, which is the time at which the CPAP Time Marker frame was sent minus the start time of a BCQF cycle. While, in principle, any cycle start time in the past or future could be chosen, the CPAP transmitter shall choose a cycle such that $-T_C \le T_I \le T_C$, where T_C is the BCQF cycle time. This time can be positive or negative, depending on whether it is compared to a past or a future BCQF cycle.

The sequence of events in the CPAP receiver, for that same CPAP sequence, is as follows:

d) (Event B1 in Figure 99-2.) The CPAP receiver receives a CPAP Time Marker Frame.
e) (Event B2 in Figure 99-2.) The CPAP obtains a value, in locally-meaningful units, for the time at which the first bit of the CPAP Time Marker Frame was received.
f) (Event B3 in Figure 99-2.) The CPAP receiver receives a CPAP Phase Offset Message containing the time offset T_I.
g) (Event B4 in Figure 99-2.) The CPAP receiver uses the time of receipt from step e) and the time offset T_I to compute the start time of a receive BCQF cycle that is aligned with the CPAP transmitter.

<< Editor's note: Unless the editor receives convincing opinions in the form of ballot comments to the contrary, he has no intention of creating state machines or C code for CPAP. His claim is that this would add complexity to the document, time to its development, and additional work for the reader, without increasing the likelihood of interoperability. >>

99.3 CPAP message transmission timing

The transmission of CPAPDUs is enabled by BcqfCpapTransmitEnable (100.1.3.3).. The interval between CPAPDU transmissions is governed, per port, by BcqfCpapTransmitPeriod (100.1.3.4). This is the nominal time between transmissions; the actual interval between successive transmissions can be as little as one-half BcqfCpapTransmitPeriod or as large as twice BcqfCpapTransmitPeriod. BcqfCpapTransmitPriority (100.1.3.5) determines at what priority the CPAPDUs are sent. Sending them using a BCQF class of service (with bandwidth reserved for them, of course) ensures that the CPAPDUs are transmitted on time and do not interfere with other time-critical traffic.

<< Editor's note. Input is encouraged on this subject. The editor does not believe that we should discuss the matter of accuracy to a depth approaching that in IEEE Std 802.1AS. Exactly what should be said, here, is therefore problematical, at present. >>

99.4 CPAP message frame formats

CPAP is a GD1Q User protocol, built on the Generic Dot1Q Protocol (Clause 98). The format of the CPAPDUs are defined in the following clauses.

99.4.1 Common CPAPDU format

The common portion of all CPAPDUs is specified in Table 99-1.

99.4.1.1 EtherType

The EtherType is of the Generic Dot1Q Protocol is specified in Table 98-2

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Table 99-1—Generic Dot1Q Data Unit Format

Field	Octet	Length
EtherType (99.4.1.1)	0	2
ProtocolId (99.4.1.2)	2	1
Version (99.4.1.3)	3	1
Message type {99.4.1.4)	4	2
Sequence number {99.4.1.5)	6	4
message dependent	10	

99.4.1.2 ProtocolId

The Protocol ID field is the value for CPAP from Table 98-3.

99.4.1.3 Version

The Version number shall be encoded as 0 on transmitted frames. The Version number shall be ignored on received frames.

99.4.1.4 Message type

The message type, as specified in Table 99-2.

Table 99-2—Message Type

Message	Message type field
Time Marker Frame	0
Phase Offset Message	1
reserved for future use	2–65 535

99.4.1.5 Sequence number

A four-octet binary number, encoded in decreasing order of numerical significance, with the numerically most-significant octet in position 6 in Table 99-1. This sequence number is used to match this TIme Marker Frame to a subsequence Phase Offset message. The first sequence number value transmitted in a Time Marker after a system reset is random, and it is incremented by one for each subsequence Phase Offset message sent. The sequence number value transmitted in a Phase Offset Message is the same as a recently-transmitted Time Marker message.

99.4.2 Time Marker format

The Time Marker CPCPDU carries no more information than the common CPAPDU format (99.4.1).

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

99.4.3 CPAP Phase Offset Message format

The format of a CPAP Phase Offset Message frame is specified in Table 99-3.

Table 99-3—CPAP Phase Offset Message format

Field	Octet	Length
Common CPAPDU format (99.4.1), message type = 1	0	10
Phase offset (99.4.3.1)	10	4

99.4.3.1 Phase offset

A four-octet signed binary number, encoded in decreasing order of numerical significance, with the numerically most-significant octet in position 10 in Table 99-3. This is the signed number of nanoseconds from the time at which the corresponding CPAP Time Marker frame was sent minus the start time of a BCQF cycle.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Insert the following Clause:

100. Bin Cyclic Queuing and Forwarding (BCQF)

100.1 BCQF managed objects

100.1.1 TcqfingressTable

A table of managed objects indexed by TcqfingressPort. There is one TcqfingressTable per Bridge component. These managed objects control the assignment of bin numbers to frames deposited in class of service queues configured for BCQF.

100.1.1.1 TcqfingressPort

Index. The Bridge port number of a port in a Bridge Component. In an end station, this is always 0.

100.1.1.2 TcqfingressEpoch

Read/write. A PTP timescale value. One per port. The start time of the slowest TcqfingressPeriod (100.1.1.4.3) on this port. This can be set as a network management operation. The CQF Phase Alignment Protocol (Clause 99) also sets this value. Default value 0.

100.1.1.3 TcqfCpapReceiveEnable

Boolean. If TRUE, CQF Phase Alignment Protocol (Clause 99) protocol data units (PDUs) are allowed to adjust TcqfingressEpoch (100.1.1.2). Default value TRUE.

100.1.1.4 TcqfIpvTable

A table of managed objects indexed by TcqfingressIndex (100.1.1.4.1).

100.1.1.4.1 TcqfingressIndex

An integer index to this entry in the TcqfIpvTable.

100.1.1.4.2 TcqfingressIpvList

Index. A list of integer internal priority value or IPV. Frames received with one of these values are processed by time-based CQF bin assignment (8.6.5.4). If an IPV is assigned (8.6.5.3), that value is used in preference to the frame's priority. No two entries in the TcqfIpvTable can contain the same priority or IPV value in their TcqfingressIpvList lists. No default value.

100.1.1.4.3 TcqfingressPeriod

Read/write. A rational numbers of seconds, The tCQF ingress window rotation period for frames received with the TcqfingressIpvList index (100.1.1.4.2). No default value.

NOTE—Using a rational number of seconds for each period, rather than a list of integer multiples of a base period, facilitates the addition or deletion of a traffic class while one or more other traffic classes are in use on the same port.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

100.1.1.4.4 TcqfDeadTime

Read/write. Integer in the range 0-100, inclusive. The percentage dead time (see 100.1.3.6.3) required by this system to ensure that all tCQF frames received on this port with this period will be transmitted from the appropriate BCQF bin. See also Y.3.5.

100.1.1.4.5 TcqfAllocableBandwidth

Read only. Integer in the range 0-100, inclusive. The percentage of the tCQF ingress window rotation period that is allocable to tCQF frames for this cycle period. This number is computed by the system based on knowledge of its current configuration and capabilities, and reflects any number of considerations (see Annex Y.1 for examples).

NOTE 1—This managed object reflects the limitations of the system. Network management and/or reservation protocols can have reason to further restrict the allocable bandwidth.

NOTE 2—This managed object does not include dead time (TcqfDeadTime, 100.1.1.4.4). If TcqfAllocableBandwidth and TcqfDeadTime add up to more than 100, then the effective allocable bandwidth is reduced.

100.1.2 BcqfStreamControlTable

The stream control table two tables, each consisting of instances of the BcqfControlTableEntry (100.1.2.4):

a) A table indexed by ingress port (100.1.2.1) and by egress port; (100.1.2.2); and
b) A table indexed by stream_handle and egress port (100.1.2.3).

If a received frame is assigned a stream_handle, and if there is an entry in the BcqfStreamControlTable for that stream_handle and egress port, then that entries is applied to the frame. If not, then the entry corresponding to the ingress and egress ports through which the frame passes apply. If the frame is forwarded on multiple ports, a different entry in the BcqfStreamControlTable applies for each egress port.

Entries in this table can be created or deleted via resource allocations protocols such as RAP (IEEE Std 802.1Qdd).

100.1.2.1 BcqfControlIngressPort

Index. The integer Bridge port number of a port in a Bridge Component on which a frame was received. In an end station, this is always 0.

100.1.2.2 BcqfControlIngressPort

Index. The integer Bridge port number of a port in a Bridge Component to which a frame is to be enqueued for transmission.

100.1.2.3 BcqfControlStreamHandle

Index. The integer stream_handle (8.6.5.3) assigned to the frame by the Frame Identification function of IEEE 802.1CB.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

100.1.2.4 BcqfControlTableEntry

100.1.2.4.1 CcqfAllocatedBits

Read/Write. The maximum number of transmission bit times per BCQF bin period allocated to this Stream or port-to-port flow. This is the bandwidth allocated the Stream or flow. The total of all CcqfAllocatedBits values over all entries for a given egress port cannot exceed the value computed from BcqfEgressPeriod (100.1.3.6.2), BcqfEgressDeadTime (100.1.3.6.3), and BcqfAllocableBandwidth (100.1.3.6.4).

100.1.2.4.2 BcqfMinimumDelay

Read-only. Integer number of nanoseconds. See Y.3.2 and Figure Y-4. This is the BCQF-caused delay. The starting point of this delay is the IEEE Std 802.3 Clause 90 receive timestamp moment for a minimum-length frame transmitted at the earliest possible moment in a cycle by the adjacent transmitting Bridge. The ending point of the minimum BCQF delay is the earliest time when the bin, into which the frame is stored, could be selected for transmission, assuming that the intentional delay (100.1.2.4.4) is 0.

For the purposes of computing the end-to-end delivery time of a frame, the system sets BcqfMinimumDelay so that network management can add the one-way link delay to BcqfMinimumDelay to get the minimum delay across this system.

<< Editor's note: Link delay variation also affects per-hop delay; This variable should include clock inaccuracies and minor link delay variations. Major link delay variation is handled, below. There are some details to be worked out. >>

NOTE 1—This value cannot be computed until BcqfEgressEpoch (100.1.3.2) and BcqfEgressEpoch (100.1.3.2) are both known. CPAP (Clause 99) can cause BcqfEgressEpoch to change, thus affecting the end-to-end delivery time.

NOTE 2—Computing a value for this variable is not trivial. See Clause Y in general, and Y.3 in particular, for an explanation of factors that influence this computation.

100.1.2.4.3 BcqfMaximumDelay

Read-only. Integer number of nanoseconds. See Y.3.2 and Figure Y-4. This is the BCQF-caused delay. The starting point of this delay is the IEEE Std 802.3 Clause 90 receive timestamp moment for a minimum-length frame transmitted at the latest possible moment in a cycle by the adjacent transmitting Bridge. The ending point of the minimum BCQF delay is the latest possible time when the bin, into which the frame is stored, could be selected for transmission.

BcqfMaximumDelay does not include:

a) Intentional delay. This is included in BcqfIntentionalDelayBins (100.1.2.4.4);
b) Delay cause by cCQF frames overflowing into a succeeding bin, because the target bin is full. The maximum additional delay of this sort is specified by BcqfMaximumExtraCcqfBins (100.1.2.4.6).
c) The time required to transmit the frame from its bin (up to BcqfEgressPeriod, 100.1.3.6.2).

NOTE—Computing a value for this variable is not trivial. See Clause Y in general, and Y.3 in particular, for an explanation of factors that influence this computation.

100.1.2.4.4 BcqfIntentionalDelayBins

Read/write. An integer number of additional BCQF cycles to be imposed on frames. in order to deliberately increase their delivery delay in this Bridge.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

This parameter is specified in order to cover two use cases, but other uses are possible:

a) This parameter can impose delays on Streams that are being replicated and sent on multiple topological paths, so that the time taken to traverse the longer and shorter paths can be equalized.
b) This parameter can be used to initialize extra buffing required for very long links, where the link delay can vary over a range comparable to the bin rotation period, and CPAP (Clause 99) is used to keep the slowly-varying ingress and Egress cycles aligned. In that case, BcqfMaximumExtraCcqfBins (100.1.2.4.6) can be used to bound the required buffer space.

100.1.2.4.5 BcqfBinSelectionMode

Read/write. Enumerated value. Default **time_based_assignment**.

a) **time_based_assignment**: The egress bin is to be assigned using time-based CQF (8.6.5.4)
b) **count_based_assignment**: The egress bin is to be assigned using count-based CQF (8.6.5.5).
c) **time_count_assignment:** The egress bin is to be assigned using time-based CQF (8.6.5.4), and checked for overflow on a per-stream basis.

100.1.2.4.6 BcqfMaximumExtraCcqfBins

Read/write. The integer maximum number of extra bins, beyond those included in BcqfMaximumDelay (100.1.2.4.3), that can be used for Streams using this table entry for count-based CQF. Default value 0.

NOTE—Because this value can apply to individual Streams, or to ingress/egress port pairs, the actual number of octets of buffer space required to accommodate a non-zero value in BcqfMaximumExtraCcqfBins (100.1.2.4.6) can be less than that required to contain a full bin of data.

100.1.3 BcqfEgressTable

A table of managed objects indexed by BcqfEgressPort (100.1.3.1). There is one BcqfEgressTable per Bridge component. These managed objects control the assignment of bin numbers to frames deposited in class of service queues configured for BCQF.

100.1.3.1 BcqfEgressPort

Index. The Bridge port number of a port in a Bridge Component.

100.1.3.2 BcqfEgressEpoch

Read/write. A PTP timescale value. One per port. The start time of the slowest BcqfEgressPeriod (100.1.3.6.2) on this port. This can be set as a network management operation. Default value 0.

100.1.3.3 BcqfCpapTransmitEnable

Read/write. Boolean. If TRUE, CQF Phase Alignment Protocol (Clause 99) protocol data units (CPAPDUs) are transmitted from this port. Default value TRUE.

100.1.3.4 BcqfCpapTransmitPeriod

Read/write. Rational number of seconds specifying the nominal interval between transmission of CPAPDUs.

100.1.3.5 BcqfCpapTransmitPriority

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

100.1.3.6 BcqfTrafficClassTable

The BcqfTrafficClassTable is indexed by class of service, and contains a list of managed objects controlling the assignment of frames to bins for a particular BCQF class of service queue.

100.1.3.6.1 BcqfClassOfService

Index. Integer class of service number (Transmission Selection Algorithm Table, 8.6.8). The index into the BcqfTrafficClassTable (100.1.3.6).

100.1.3.6.2 BcqfEgressPeriod

Read/write. A rational number of seconds. The BCQF bin rotation period (T_C, 8.6.8.7) for BCQF frames transmitted through the traffic class BcqfClassOfService (100.1.3.6.1). No default value.

NOTE—Using a rational number of seconds for each traffic class, rather than a list of integer multiples of a base period, facilitates the addition or deletion of a traffic class while one or more other traffic classes are in use on the same port.

100.1.3.6.3 BcqfEgressDeadTime

Read/write. Integer in the range 0-100, inclusive. The dead time percentage. No part of any frame is transmitted from this traffic class within this percentage of the last part of the bin rotation periods. (See “T_D” in Y.3.2 for a non-normative description of this parameter.)

NOTE—Dead time for one class of service can be used for transmitting frames from lower-numbered classes of service.

100.1.3.6.4 BcqfAllocableBandwidth

Read only. Integer in the range 0-100, inclusive. The percentage of the BCQF bin rotation period that is allocable to BCQF frames for this class of service. This number is computed by the system based on knowledge of its current configuration and capabilities, and reflects considerations such as, but not limited to, output delay, link delay, and clock accuracy (See T_V in Y.3.2 for examples).

NOTE 1—This managed object reflects the limitations of the system. Network management and/or reservation protocols can have reason to further restrict the allocable bandwidth.

NOTE 2—This managed object does not include dead time (BcqfEgressDeadTime, 100.1.3.6.3). If BcqfAllocableBandwidth and BcqfEgressDeadTime add up to more than 100, then the effective allocable bandwidth is reduced.

100.1.3.6.5 BcqfMaximumFanIn

Read/write. Integer. Maximum number of ingress ports that can forward BCQF frames to this class of service on this port for transmission. This can be used by the Bridge to compute BcqfMinimumDelay (100.1.2.4.2). The Bridge is not required to verify that this parameter is not exceeded by the actual traffic flowing through the Bridge.

<< Editor’s note: If little or no fan-in is allowed, then a bin can start transmitting while it is still open for receiving frames from ingress ports, thus improving the per-hop latency. Suggestions are welcome on how it can be enforced properly. Without enforcement, it is dangerous. >>

100.1.4 BCQF configuration consistency

A system shall not allow any of the following constraints on managed objects to be violated:

a) If more than ore traffic class on a Bridge Port is configured for BCQF, the lowest-numbered such traffic class (the least urgent) has the largest value of BcqfEgressPeriod (100.1.3.6.2).

b) The BcqfEgressPeriod (100.1.3.6.2) value for each traffic class is an integer multiple of the next-lower-numbered traffic class’s BcqfEgressPeriod value.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

c) The sum, over all entries in the BcqfStreamControlTable (100.1.2) for any given egress port, including entries for stream_handles that can be transmitted on that port, of the CcqfAllocatedBits (100.1.2.4.1), cannot exceed the maximum number of allocable bit times computed from BcqfEgressPeriod (100.1.3.6.2), BcqfEgressDeadTime (100.1.3.6.3), and BcqfAllocableBandwidth (100.1.3.6.4).

d) If more than ore class of service on a Bridge Port is configured for BCQF, the lowest-numbered such traffic class (the least urgent) has the largest value of BcqfEgressPeriod (100.1.1.4).

e) The BcqfEgressPeriod (100.1.3.6.2) value for each traffic class is an integer multiple of the next-lower-numbered traffic class's BcqfEgressPeriod value.

100.2 BCQF LLDP TLVs

<< Editor's note: In general, it would be good for two BCQF devices to exchange information to allow them to verify that they are both configured with the same priority levels, T_C values, etc. LLDP seems a reasonable choice for this. Comments/suggestions are welcome. >>

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Annex A

(normative)

PICS proforma—Bridge implementations[2]

<< Editor’s note: No entries for the PICS for BCQF have been generated, yet. The changes to Annex A in this draft of the amendment are all due to changing the name of the existing CQF to “sCQF”. >>

[2] *Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

A.5 Major capabilities

Change the following in Table A.5:

Item	Feature	Status	References	Support
ATS	Does the implementation support Asynchronous Traffic Shaping?	O	5.4.1.10, 5.13.1.3, 8.6.5.2.2, 8.6.6 items d) and e), 8.6.8, 8.6.8.5, 8.6.11, 12.31	Yes [ ] No [ ]
~~CQF~~sCQF	Does the implementation support scheduled cyclic queuing and forwarding?	O	5.4.1.9.1, 5.13.1.2.1	Yes [ ] No [ ]
YANG	Does the implementation support management operations using the YANG modules?	MGMT:O	8.12, 48	Yes [ ] No [ ] N/A [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.14 Bridge management

Change the following rows of Table A.14:

Item	Feature	Status	References	Support
MGT-248	Does the implementation support the management entities defined in 12.29?	SCHED OR ~~CQF~~sCQF: M	5.4.1 item ad), 5.4.1.9.1 item c), 5.13.1.2.1 item c), 12.29	Yes [ ] N/A [ ]
MGT-249	Does the implementation support the management entities defined in 12.30?	PRE: M	5.4.1 item ae), 12.30	Yes [ ] N/A [ ]
MGT-250	Does the implementation support the management entities defined in 12.31 for PSFP?	PSFP OR ~~CQF~~sCQF: M	5.4.1.9.1 item e), 5.13.1.2.1 item e), 8.6.5.2.1, 8.6.10, 12.31	Yes [ ] N/A [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.24 Management Information Base (MIB)

Change the following in Table A.24:

Item	Feature	Status	References	Support
MIB-1	Is the IEEE8021-SPB-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND SPB:O	5.4.5, 17.7.19	Yes [ ] No [ ] N/A [ ]
MIB-2	Are the IEEE8021-ECMP-MIB module objects ieee8021EcmpEctStaticTable and ieee8021EcmpTopSrvTable supported?	MIB AND ECMP:O	5.4.5.1, 17.7.10	Yes [ ] No [ ] N/A [ ]
MIB-3	Is the IEEE8021-ECMP-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND FF:O	5.4.5.2, 17.7.21	Yes [ ] No [ ] N/A [ ]
MIB-4	Are PCR objects in the IEEE8021-SPB-MIB supported per ieee8021PcrCompliance ?	MIB AND PCR:O	5.5, 17.7.19	Yes [ ] No [ ] N/A [ ]
MIB-5	Is the IEEE8021-ST-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND (SCHED OR ~~CQF~~sCQF): O	5.4.1 item ad), 5.4.1.9.1 item c), 5.13.1.2.1 item c), 12.29, 17.7.22	Yes [ ] No [ ] N/A [ ]
MIB-6	Is the IEEE8021-Preemption-MIB module fully supported (per its MODULE-COMPLIANCE)?	PRE: O	5.4.1 item ae), 12.30, 17.7.23	Yes [ ] No [ ] N/A [ ]
MIB-7	Is the IEEE8021-PSFP-MIB module fully supported (per its MODULE-COMPLIANCE)?	PSFP OR ~~CQF~~sCQF: O	5.4.1.9.1 item e), 5.13.1.2.1 item e), 8.6.5.2, 8.6.10, 12.31, 17.7.24	Yes [ ] No [ ] N/A [ ]
MIB-45	Is the IEEE8021-PBBN-AA-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND (AAB OR AAD):O	17.2.26	Yes [ ] No [ ] N/A [ ]
MIB-46	Are the Auto Attach objects in IEEE8021-LLDP-EXT-DOT1-EVB-EXTENSIONS-MIB module supported (per lldpXdot1AaCompliance MODULE-COMPLIANCE)?	MIB AND (AAB OR AAD):O	D.5.6	Yes [ ] No [ ] N/A [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

1

A.44 Scheduled traffic

Change the following rows of Table A.44:

Item	Feature	Status	References	Support
	If neither scheduled traffic (SCHED in A.5) nor scheduled cyclic queuing and forwarding (~~CQF~~sCQF in A.5) are supported, mark N/A and ignore the remainder of this table.		5.4.1, 5.13.1, 8.6.8, 8.6.9, 12.29, 17.7.22	N/A [ ]
SCHED1	Does the implementation support the state machines and associated definitions specified in 8.6.9	SCHED OR ~~CQF~~sCQF: M	5.4.1, 5.13.1, 8.6.8, 8.6.9	Yes [ ] N/A [ ]
SCHED2	Does the implementation support the management entities defined in 12.29?	SCHED OR ~~CQF~~sCQF: M	5.4.1 item ad), 5.4.1.9.1 item c), 5.13.1.2.1 item c), 12.29	Yes [ ] N/A [ ]
SCHED3	Is the IEEE8021-ST-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND (SCHED OR ~~CQF~~sCQF) :O	5.4.1 item ad), 5.4.1.9.1 item c), 5.13.1.2.1 item c), 12.29, 17.7.22	Yes [ ] N/A [ ] No [ ]

2

A.46 Per-Stream Filtering and Policing

Change the following rows of Table A.46:

Item	Feature	Status	References	Support
	If neither Per-Stream Filtering and Policing (PSFP in A.5) nor scheduled cyclic queuing and forwarding (~~CQF~~sCQF in A.5) are supported, mark N/A and ignore the remainder of this table.		5.4.1.9.1, 5.13.1.2.1, 8.6.5.2, 8.6.10, 12.31, 17.7.24	N/A[ ]
PSFP1	Does the implementation support the state machines and associated definitions as specified in 8.6.10?	PSFP OR ~~CQF~~sCQF: M	5.4.1.9.1 item b), 5.13.1.2.1 item b), 8.6.5, 8.6.10	Yes [ ] N/A [ ]
PSFP2	Does the implementation support the management entities defined in 12.31?	PSFP OR ~~CQF~~sCQF: M	5.4.1.9.1 item e), 5.13.1.2.1 item e), 8.6.5.2, 8.6.10, 12.31	Yes [ ] N/A [ ]
PSFP3	Is the IEEE8021-PSFP-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND (PSFP OR CQF):O	5.4.1.9.1 item e), 5.13.1.2.1 item e), 12.31, 17.7.24	Yes [ ] N/A [ ] No [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.47 YANG

Change the following in Table A.47:

Item	Feature	Status	References	Support
	If item YANG is not supported, mark N/A			N/A [ ]
YANG-802-TYPES	Is the *ieee802-types* module supported?	M	48.6.1	Yes [ ]
YANG-Q-TYPES	Is the *ieee802-dot1q-types* module supported?	M	48.6.2	Yes [ ]
YANG-TSN-TYPES	Is the *ieee802-dot1q-tsn-types* module supported?	O	48.6.3	Yes [ ] No [ ]
YANG-QBRIDGE	Is the *ieee802-dot1q-bridge* module supported?	M	48.6.4	Yes [ ] N/A [ ]
YANG-TPMR	Is the *ieee802-dot1q-tpmr* module supported?	TPMR:O	48.6.5	Yes [ ] No [ ] N/A [ ]
YANG-PB	Is the *ieee802-dot1q-pb* module supported?	PB:O	48.6.6	Yes [ ] No [ ] N/A [ ]
YANG-CFM-TYPES	Is the *ieee802-dot1q-cfm-types* module supported?	CFM:O	48.6.7	Yes [ ] No [ ] N/A [ ]
YANG-CFM	Is the *ieee802-dot1q-cfm* module supported?	CFM:O	48.6.8	Yes [ ] No [ ] N/A [ ]
YANG-CFM-BRIDGE	Is the *ieee802-dot1q-cfm-bridge* module supported?	CFM:O	48.6.9	Yes [ ] No [ ] N/A [ ]
YANG-CFM-ALARM	Is the *ieee802-dot1q-cfm-alarm* module supported?	CFM:O	48.6.10	Yes [ ] No [ ] N/A [ ]
YANG-STREAMS	Is the *ieee802-dot1q-stream-filters-gates* module supported?	ATS:O	48.6.11	Yes [ ] No [ ] N/A [ ]
YANG-STREAMS-BRIDGE	Is the *ieee802-dot1q-stream-filters-gates-bridge* module supported?	ATS:O	48.6.12	Yes [ ] No [ ] N/A [ ]
YANG-ATS	Is the *ieee802-dot1q-ats* module supported?	ATS:O	48.6.13	Yes [ ] No [ ] N/A [ ]
YANG-ATS-BRIDGE	Is the *ieee802-dot1q-ats-bridge* module supported?	ATS:O	48.6.14	Yes [ ] No [ ] N/A [ ]
YANG-CI	Is the *ieee802-dot1q-congestion-isolation* module supported?	CI:O	48.6.15	Yes [ ] No [ ] N/A [ ]
YANG-CI-BRIDGE	Is the *ieee802-dot1q-congestion-isolation-bridge* module supported?	CI:O	48.6.16	Yes [ ] No [ ] N/A [ ]
YANG-SCHED	Is the *ieee802-dot1q-sched* module supported?	SCHED or ~~CQF~~sCQF:O	48.6.17	Yes [ ] No [ ] N/A [ ]
YANG-SCHED-BRIDGE	Is the *ieee802-dot1q-sched-bridge* module supported?	SCHED or ~~CQF~~sCQF:O	48.6.18	Yes [ ] No [ ] N/A [ ]
YANG-PREEMP	Is the *ieee802-dot1q-preemption* module supported?	PRE:O	48.6.19	Yes [ ] No [ ] N/A [ ]
YANG-PREEMP-BRIDGE	Is the *ieee802-dot1q-preemption-bridge* module supported?	PRE:O	48.6.20	Yes [ ] No [ ] N/A [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.47 YANG *(continued)*

Change the following in Table A.47:

YANG-PSFP	Is the *ieee802-dot1q-psfp* module supported?	PSFP or ~~CQF~~sCQF:O	48.6.21	Yes [ ] No [ ] N/A [ ]
YANG-PSFP-BRIDGE	Is the *ieee802-dot1q-psfp-bridge* module supported?	PSFP or ~~CQF~~sCQF:O	48.6.22	Yes [ ] No [ ] N/A [ ]
YANG-LLDP-PBBN-AA	Is the *ieee802-dot1q-lldp-pbbn-aa-tlv* module supported?	YANG AND (AAB OR AAD): O	D.6.6.7	Yes [ ] No [ ] N/A [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.48 Stream reservation remote management (SRRM)

Item	Feature	Status	References	Support
	If Stream reservation remote management functionality (SRRM of A.5) is not supported, mark N/A and ignore the remainder of this table.			N/A [ ]
SRRM-1	What management protocol standard(s) or specification(s) are supported (server side)?	M	5.4.1.11(b)	____________ ____________ ____________
SRRM-2	Does the implementation report delay through the Bridge?	M	12.32.1	Yes [ ]
SRRM-3	Does the implementation report propagation delay?	M	12.32.2	Yes [ ]
SRRM-4	Does the implementation support Static Trees for static configuration of spanning trees?	M	5.4.1.11 item c), 12.32.3	Yes [ ]
SRRM-5	Does the implementation support MRP External Control for the MSRP application?	O SRP: M	12.32.4	Yes [ ] No [ ] N/A [ ]
SRRM-6	Does the implementation support MRP External Control for the MVRP application?	O MVRP: M	12.32.4	Yes [ ] No [ ] N/A [ ]
SRRM-7	Does the implementation support MRP External Control for the MMRP application?	O MMRP: M	12.32.4	Yes [ ] No [ ] N/A [ ]
SRRM-8	Does the implementation support queue reservation for traffic classes using the strict priority algorithm, through configuration of adminIdleSlope?	M	5.4.1.11 item e), 12.20.1, 34.3	Yes [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.49 TSN Centralized Network Configuration (CNC) station

Item	Feature	Status	References	Support
	If the functionality of a Centralized Network Configuration station (CNC-S of A.5) is not supported, mark N/A and ignore the remainder of this table.			N/A [ ]
CNC-S-1	What management protocol standard(s) or specification(s) are supported (client side)?	M	5.29 item a)	______________ ______________ ______________
CNC-S-2	Does the implementation support the managed object definitions and encodings for Stream reservation remote management?	M	5.29 item b), 12.32	Yes [ ]
CNC-S-3	Does the implementation support the managed object definitions and encodings for scheduled traffic?	O	12.29	Yes [ ] No [ ]
CNC-S-4	Does the implementation support the managed object definitions and encodings for frame preemption?	O	12.30	Yes [ ] No [ ]
CNC-S-5	Does the implementation support the managed object definitions and encodings for IEEE Std 802.1AS?	O	IEEE Std 802.1AS	Yes [ ] No [ ]
CNC-S-6	Does the implementation support the managed object definitions and encodings for IEEE Std 802.1CB?	O	IEEE Std 802.1CB	Yes [ ] No [ ]
CNC-S-7	Does the implementation support MRP External Control for the MSRP application?	O	12.32.4	Yes [ ] No [ ]
CNC-S-8	Does the implementation support MRP External Control for the MVRP application?	O	12.32.4	Yes [ ] No [ ]
CNC-S-9	Does the implementation support MRP External Control for the MMRP application?	O	12.32.4	Yes [ ] No [ ]
CNC-S-10	What user/network configuration protocol standard(s) or specification(s) are supported?	M	5.29 item c), 46.2.2	______________ ______________ ______________
CNC-S-11	Does the implementation conform to the conditional requirements for use of a YANG-based protocol?	M	5.29 item d), 46.2, 46.3	Yes [ ]
CNC-S-13	Does the implementation conform to the conditional requirements for use of SRP?	M	5.29 item e), 46.2, 12.32.4	Yes [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.50 VDP for NVO3 nNVE Devices

Item	Feature	Status	References	Support
	If VDP-NVO3 nNVE functionality (VDP-NVO3-nNVE in A.5) is not supported, mark N/A and ignore the remainder of this table.			N/A [ ]
VDP-nNVE-1	Does the implementation support the Bridge role of VDP on each SBP?	M	5.30.1, Clause 41	Yes [ ]
VDP-nNVE-2	Does the implementation support the Bridge VDP state machine as specified in Clause 41?	M	Clause 41, 41.5.2	Yes [ ]
VDP-nNVE-3	Does the implementation support assignment of VIDs to GroupIDs?	M	5.30.1, 41.2.9	Yes [ ]
VDP-nNVE-4	Does the implementation support at least one SBP on the nNVE?	M	5.30.1, Clause 40	Yes [ ]
VDP-nNVE-5	Does the implementation support an LLDP nearest Customer Bridge database including the EVB TLV on each SBP?	O	5.30.1, D.2.12	Yes [ ] No [ ]
VDP-nNVE-6	Does the implementation support the EVB status parameters for EVBMode = NVO3 and NVERole = nNVE for the nNVE role?	O	5.30.1, 40.4, 40.5	Yes [ ] No [ ]
VDP-nNVE-7	Does the implementation support the EVB Bridge status parameters for IPv4 address capability?	VDP-nNVE-6:M	D.2.12.3.5	Yes [ ] No [ ]
VDP-nNVE-8	Does the implementation support the EVB Bridge status parameters for IPv6 address capability?	VDP-nNVE-6:M	D.2.12.3.4	Yes [ ] No [ ]
VDP-nNVE-9	Does the implementation support ECP on each SBP?	O	5.30.1, Clause 43	Yes [ ] No [ ]
VDP-nNVE-10	Does the implementation support the use of M, S, and N bits in VDP?	O	5.30.1, 41.2.3	Yes [ ] No [ ]
VDP-nNVE-11	Does the implementation support the use of IPv4 addresses in VDP filter info format?	O	5.30.1, 41.2.9	Yes [ ] No [ ]
VDP-nNVE-12	Does the implementation support the use of IPv6 addresses in VDP filter info format?	O	5.30.1, 41.2.9	Yes [ ] No [ ]
VDP-nNVE-13	Does the implementation support an LLDP database addressed by a unicast MAC address including the EVB TLV on each SBP?	O	5.30.1, D.2.12	Yes [ ] No [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.51 VDP for NVO3 tNVE Devices

Item	Feature	Status	References	Support
	If VDP-NVO3 tNVE functionality (VDP-NVO3-tNVE in A.5) is not supported, mark N/A and ignore the remainder of this table.			N/A [ ]
VDP-tNVE-1	Does the implementation support the Station role of VDP on each URP?	M	5.30.1, Clause 41	Yes [ ]
VDP-tNVE-2	Does the implementation support the Station VDP state machine as specified in Clause 41?	M	Clause 41, 41.5.3	Yes [ ]
VDP-tNVE-3	Does the implementation support an LLDP nearest Customer Bridge database including the EVB TLV on each URP?	O	5.30.2, D.2.12	Yes [ ] No [ ]
VDP-tNVE-4	Does the implementation support the EVB status parameters for EVBMode = NVO3 and NVERole = tNVE for the tNVE role?	O	5.30.2, 40.4, 40.5	Yes [ ] No [ ]
VDP-tNVE-5	Does the implementation support the EVB Bridge status parameters for IPv4 address capability?	VDP-tNVE-4:M	D.2.12.4.5	Yes [ ] No [ ]
VDP-tNVE-6	Does the implementation support the EVB Bridge status parameters for IPv6 address capability?	VDP-tNVE-4:M	D.2.12.4.4	Yes [ ] No [ ]
VDP-tNVE-7	Does the implementation support ECP on each URP?	O	5.30.2, Clause 43	Yes [ ] No [ ]
VDP-tNVE-8	Does the implementation support the use of M, S, and N bits in VDP?	O	5.30.2, 41.2.3	Yes [ ] No [ ]
VDP-tNVE-9	Does the implementation support the use of IPv4 addresses in VDP filter info format?	O	5.30.2, 41.2.9	Yes [ ] No [ ]
VDP-tNVE-10	Does the implementation support the use of IPv6 addresses in VDP filter info format?	O	5.30.2, 41.2.9	Yes [ ] No [ ]
VDP-tNVE-11	Does the implementation support an LLDP database addressed by a unicast MAC address including the EVB TLV on each URP?	O	5.30.2, D.2.12	Yes [ ] No [ ]
VDP-tNVE-12	Does the RRREQ in EVB station status parameters set to FALSE to make reflective relay always disabled?	VDP-tNVE-4:M	D.2.12.4	Yes [ ] No [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

A.52 Asynchronous Traffic Shaping

Item	Feature	Status	References	Support
	If Asynchronous Traffic Shaping (ATS in A.5) is not supported, mark N/A and ignore the remainder of this table.		5.4.1.10, 5.13.1.3, 8.6.5.2.2, 8.6.6 items d) and e), 8.6.8.5, 8.6.8, 8.6.8.5, 8.6.11, 12.31	N/A [ ]
ATS-1	Does the implementation support the ATS per-stream classification and metering for ATS as specified in 8.6.5.2.2?	ATS:M	5.4.1.10, 5.13.1.3, 8.6.5.2.2	Yes [ ]
ATS-2	Does the implementation support the ATS transmission selection algorithm as specified in 8.6.8.5?	ATS:M	5.4.1.10, 5.13.1.3, 8.6.8.5	Yes [ ]
ATS-3	Does the implementation support the ATS scheduler state machines as specified in 8.6.11?	ATS:M	5.4.1.10, 5.13.1.3, 8.6.11	Yes [ ]
ATS-4	Does the implementation support the management entities defined in 12.31 for ATS?	ATS:M	5.4.1.10, 5.13.1.3, 12.31	Yes [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Annex B

(normative)

PICS proforma—End station implementations[3]

<< Editor's note: No entries for the PICS for BCQF have been generated, yet. The changes to Annex B in this draft of the amendment are all due to changing the name of the existing CQF to "sCQF". >>

B.5 Major capabilities

Change the following in Table B.5:

Item	Feature	Status	References	Support	
PRE	Does the implementation support frame preemption?	O	5.4.1, 5.13.1, 6.7.2, 8.6.8, 12.30, 17.7.23	Yes [ ]	No [ ]
PSFP	Does the implementation support PSFP?	O	8.6.5.2.1, 8.6.10, 12.31	Yes [ ]	No [ ]
ATS	Does the implementation support Asynchronous Traffic Shaping?	O	8.6.5.2.2, 8.6.8, 8.6.8.5, 8.6.11, 12.31	Yes [ ]	No [ ]
~~CQF~~sCQF	Does the implementation support scheduled cyclic queuing and forwarding?	O	5.25, 5.28.1	Yes [ ]	No [ ]
CNC-S	Does the implementation support Centralized Network Configuration (CNC) station functionality?	O	5.29, 46.2, A.17	Yes [ ]	No [ ]
CI-S	Does the implementation support the functionality of a Congestion Isolation?	O	5.32, Clause 49	Yes [ ]	No [ ]
AAD	Does the implementation support AAD functionality?	O	5.33, Clause 50	Yes [ ]	No [ ]

B.15 Scheduled traffic

Change the following in Table B.5:

Item	Feature	Status	References	Support
	If neither scheduled traffic (SCHED in B.5) nor scheduled cyclic queuing and forwarding (~~CQF~~sCQF in B.5) are supported, mark N/A and ignore the remainder of this table.		5.25, 5.28.1, 8.6.8, 8.6.9, 12.29, 17.7.22	N/A [ ]
SCHED1	Does the implementation support the state machines and associated definitions specified in 8.6.9?	SCHED OR ~~CQF~~sCQF: M	5.28.1 item b), 8.6.8, 8.6.9	Yes [ ] N/A [ ]

[3] *Copyright release for PICS proformas:* Users of this standard may freely reproduce the PICS proforma in this annex so that it can be used for its intended purpose and may further publish the completed PICS.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

B.15 Scheduled traffic

Change the following in Table B.5:

Item	Feature	Status	References	Support
SCHED2	Does the implementation support the management entities defined in 12.29?	SCHED OR ~~CQF~~sCQF: M	5.28.1 item c), 12.29	Yes [ ] N/A [ ]
SCHED3	Is the IEEE8021-ST-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND (SCHED OR ~~CQF~~sCQF): O	5.28.1 item c), 12.29, 17.7.22	Yes [ ] No [ ] N/A [ ]
SCHED4	Is the *ieee802-dot1q-sched* YANG module supported?	YANG AND (SCHED OR ~~CQF~~sCQF): O	5.28.1 item c), 12.29, 48.6.17	Yes [ ] No [ ] N/A [ ]

B.17 Per-Stream Filtering and Policing

Change the following in Table B.17:

Item	Feature	Status	References	Support
	If neither Per-Stream Filtering and Policing (PSFP in B.5) nor scheduled cyclic queuing and forwarding (~~CQF~~sCQF in B.5) are supported, mark N/A and ignore the remainder of this table.		5.28.1 items d) and e), 8.6.5.2.1, 8.6.10, 12.31, 17.7.24	N/A[ ]
PSFP1	Does the implementation support the state machines and associated definitions as specified in 8.6.10?	PSFP OR ~~CQF~~sCQF: M	5.28.1 items b) and d), 8.6.5.3, 8.6.10,	Yes [ ] N/A [ ]
PSFP2	Does the implementation support the management entities defined in 12.31 for PSFP?	PSFP OR ~~CQF~~sCQF: M	5.28.1 item e), 8.6.5.2, 8.6.10, 12.31	Yes [ ] N/A [ ]
PSFP3	Is the IEEE8021-PSFP-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND (PSFP OR ~~CQF~~sCQF): O	12.31, 17.7.24	Yes [ ] No [ ] N/A [ ]
PSFP4	Is the *ieee802-dot1q-psfp* module supported?	YANG AND (PSFP OR ~~CQF~~sCQF): O	12.31, 48.6.21	Yes [ ] No [ ] N/A [ ]

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Annex T

(informative)

Scheduled Cyclic queuing and forwarding (sCQF)[4]

T.1 Overview of ~~CQF~~sCQF

Cyclic queuing and forwarding (~~CQF~~sCQF) is a method of traffic shaping that can deliver deterministic, and easily calculated, latency for time-sensitive traffic streams. As the name implies, the principle underlying ~~CQF~~sCQF is that stream traffic is transmitted and queued for transmission along a network path in a cyclic manner. Time is divided into numbered time intervals i, i+1, i+2, ... i+N, each of duration d. Frames transmitted by a Bridge, *Alice*, during time interval i are received by a downstream Bridge, *Bob*, during time interval i and are transmitted onwards by *Bob* towards Bridge *Charlie* during time interval i+1, and so on. A starting assumption is that, for a given traffic class, all Bridges and all end stations connected to a given bridge have a common understanding (to a known accuracy) of the start time of cycle i, and the cycle duration, d.

Frames transmitted by *Alice* during interval i are transmitted by *Bob* in interval i+1; the maximum possible delay experienced by a given frame is from the beginning of i to the end of i+1, or twice d. Similarly, the minimum possible delay experienced is from the end of i to the beginning of i+1, which is zero. More generally, the maximum delay experienced by a given frame is:

$$(h+1) \times d$$

and the minimum delay experienced by a given frame is:

$$(h-1) \times d$$

where h is the number of hops.

This illustrates the attraction of ~~CQF~~sCQF as a technique for handling time-sensitive traffic; the latency introduced as a frame transits the network is completely described by the cycle time and the number of hops, and is unaffected by any other topology considerations, including interference from other non time-sensitive traffic. This only holds, however, if frames are kept to their allotted cycles; if, for example, some of the frames that were expected to be received by *Bob* during cycle i do not appear until cycle i+1 has started, then the stated assumptions about maximum latency calculation no longer hold. Careful choice of cycle times, alignment of cycle times among the Bridges in the network, and the timing of first and last transmissions within a cycle are required in order to ensure that the desired latency bounds are achieved.

Any delays through a particular intermediate relay (for example, *Bob*) do not affect the end-to-end delay so long as *Bob's* performance does not affect the correct assignment of frames to time intervals.

Since one of the goals for the handling of time-sensitive streams is zero frame loss (assuming that no unrecognizable non-conformant traffic is present), it is prudent to assume that reception is continuous—a frame received by a downstream system will always be assigned to one interval or another. This places most of the burden of correct interval assignment on the transmitting system; frames should not be transmitted if incorrect interval assignment is possible upon reception. It is therefore necessary to define the anticipated (and accommodated) errors in reception assignment with respect to the point in interval time, t, where interval i–1 becomes interval i. A relay (such as *Bob*) can of course choose when to start reception assignment to i in relation to t; it is assumed that *Bob's* intent is that the earliest frame to be assigned to i is the first whose very last octet (or other frame transmission encoding symbol) is still on the transmission medium (or other definable external event to what is considered to be *Bob* reference point) at t, thus placing any accommodation of known implementation dependent delays within *Bob* under *Bob's* control.

[4] In early discussions, ~~CQF~~sCQF was known as the "Peristaltic Shaper" [B67].

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

While *Bob* attempts to start *i* reception with a frame coming off the medium at *i*, and may factor known and repeatable internal delays into the way he goes about that intent, his actual start time depends on:

a) The error in *Bob's* time sync (i.e., the error in his determination as to when *t* actually occurs).
b) The maximum deviation (jitter) in *Bob's* use of that time.
c) Additional delays that *Bob* does not account for, such as delays in selecting the output queue to be used for *i.*

Alice has to stop transmitting frames for *i*–1 before *t*, by a time that is the sum of *Bob's* possible early start of *i* as a consequence of a) through c), and the following:

d) The error in *Alice's* time sync (i.e., the error in her determination as to when *t* occurs).
e) The maximum deviation (jitter) in *Alice's* use of that time.
f) The time between *Alice* deciding to commit a frame for transmission and the appearance of the last octet/symbol "on the medium" at *Alice's* end.
g) The length of "the medium" in transmission time, i.e., the time for the last octet/symbol to leave *Alice* and reach *Bob*, including any consideration of the effect of interfering frames or fragments.

The description of ~~CQF~~sCQF in terms of a number of consecutive intervals (as opposed to their support by "odd/even" queues, as discussed in T.2 onwards) gives easy answers to what to do with traffic still queued when its selected transmission interval has expired—discard it, or mark it down (discard eligible or priority change) and generate an alarm. In an environment where the stream bandwidth is allocated appropriately (i.e., the bandwidth allocated per time interval is less than can be received/transmitted in the chosen interval duration), this will be a rare occurrence, the traffic that follows will be conformant, and the overall system performance will be recoverable.

The discussion so far has assumed that all link speeds are the same; however, the situation becomes more complicated when links of different speeds are considered. One typical arrangement might comprise low speed links at the start and end of the path (network periphery to periphery), another with the high speed towards one end (periphery to core or vice versa). Taking the first of these, and placing *Alice* at the first transition from slow to fast, *Bob* as her fast neighbor, *Charlie* as his fast neighbor, and *Donald* at the transition from fast to slow, the important thing (treating the fast core of the network as ~~a CQF~~an SCQF black box) is that all conformant traffic received by *Alice* in interval *i* (say) is transmitted by *Donald* in a later interval *i*+*n*. A number of internal arrangements might be made between *Alice*, *Bob*, *Charlie*, and *Donald* to make this happen and would be valid from an external ~~CQF~~sCQF perspective. It is also possible to consider fractional *n*, where *n* is still > 1, as *Alice* may need to collect the entirety of any slow cycle before transmitting that in a more compressed burst into the rest of the fast network. More complex possibilities are equivalent to redefining the slow cycle time. Some of the less elaborate possibilities for the use of links of different speeds are discussed in T.5.

T.2 An approach to ~~CQF~~sCQF implementation

In essence, the approach involves the use of two transmission queues and a cycle timer. During even numbered cycles (intervals), queue 1 accumulates received frames from the Bridge's reception Ports (and does not transmit them), while queue 2 transmits any queued frames from the previous odd-numbered cycle (and does not receive any frames). During odd-numbered cycles, queue 2 accumulates received frames from the Bridge's reception Ports (and does not transmit them), while queue 1 transmits any queued frames from the previous even-numbered cycle (and does not receive any frames). With appropriate choice of receive and transmit cycle times (see T.5), such that, for any given stream, the cycle is at least long enough to accommodate all of the time-sensitive traffic that will need to be transmitted on the Bridge Port during the class measurement interval for that stream (see 34.6.1.1, also known as the observation interval in IEEE Std 802.1BA™ [B12]), plus a maximum-sized interfering frame (or frame fragment, if preemption is supported), then all of the stream's traffic will be accumulated during the cycle time in queues that are in receive mode, and it will all be transmitted during the cycle time when the queues switch to transmit mode.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

~~CQF~~sCQF is implemented by configuring a combination of the stream gate control mechanisms defined in 8.6.5.3 and the traffic scheduling mechanisms defined in 8.6.8.4 and 8.6.9. Per-stream filtering is used to direct received frames to one of a pair of outbound queues on a timed basis, determined by the cycle time of the per-stream filter, and traffic scheduling is used to ensure that frames are transmitted from the appropriate queue using the same cycle time, as described in the rest of this annex.

T.3 Use of Per-Stream Filtering and Policing for ~~CQF~~sCQF

The first step in establishing the filtering and queuing structures needed for ~~CQF~~sCQF is to set up one or more stream filters (8.6.5.3) and a stream gate instance (8.6.5.3) that will be receiving incoming time-sensitive frames. The stream filter(s) are configured so that all time-sensitive frames received on a given Port are directed to the same stream gate instance; in turn, the stream gate instance is configured so that the internal priority value (IPV) associated with the time-sensitive frames will direct them to one of two outbound queues on a timed basis. The use of the IPV allows this direction of frames to outbound queues to be independent of the received priority, and also does not affect the priority associated with the frame on transmission.

T.3.1 Stream filter configuration

The simplest stream filter configuration would be achieved where the same priority is used for all time-sensitive frames (and this priority is not used for any other frames); for example, the default priority assigned to SR class A (see Clause 34) could be used, in which case, the priority associated with the time-sensitive frames would be 3. The parameters that would define the stream filter for the time-sensitive frames would then be as follows:

a) The *stream_handle specification* would take the wild-card value.
b) The *priority specification* would take the priority value 3.
c) The *stream gate instance identifier* would take the value of the instance identifier for the stream gate (T.3.2).
d) In the simplest case, there would be no further per-stream classification and metering operations (8.6.5.2); however, these could be added as appropriate, for example if the maximum SDU size (8.6.5.3.1) for the time-sensitive traffic is bounded at a value less than the maximum SDU size for the medium.

This stream filter configuration results in all frames that carry a priority value of 3 being submitted to the stream gate. As the operation of stream filters is such that received frames that do not match a stream filter are handled as if subsequent per-stream classification and metering operations were not implemented, there is no need for further stream filter configuration to handle frames that carry priorities other than 3 unless there are other filtering or gating decisions that need to be taken for such frames.

T.3.2 Stream gate configuration

The *stream gate instance* (8.6.5.3) needed to support the stream filter described in T.3.1 has a *stream gate control list* that contains two entries, each containing a SetGateAndIPV operation, with parameters as follows:

1) StreamGateState = *open*, IPV = 7, TimeInterval = T
2) StreamGateState = *open*, IPV = 6, TimeInterval = T

This control list has the effect of directing any traffic that passes the stream filter specified in T.3.1 to one of two different outbound queues (assuming that the outbound Ports support 8 queues, and that the default assignments for priorities to traffic classes follows the recommendation shown in Table 34-1); in the first time interval T, traffic is directed to queue 7, in the second time interval T, to queue 6, in the third time

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

interval to queue 7, in the fourth time interval, to queue 6, and so on. The choice of time interval T is discussed in T.5; the cycle time (OperCycleTime, see 8.6.9.4.19) for the stream gate state machines would need to be set to 2T in order to accommodate the sum of the time intervals for the two gate operations. See Figure T-1.

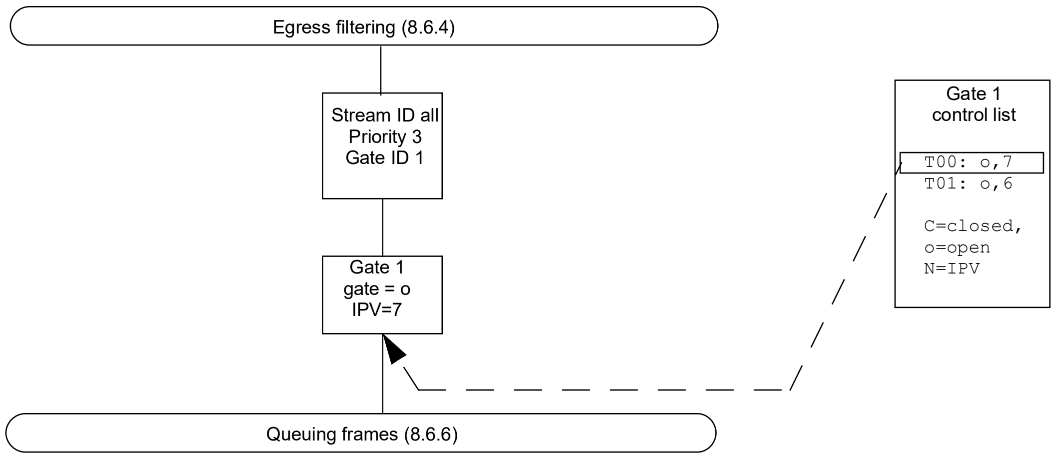


Figure T-1—Example Stream Filer and Stream Gate configuration for ~~CQF~~sCQF

T.4 Use of traffic scheduling for ~~CQF~~sCQF

The traffic scheduling support needed on each outbound Port in order to support the PSFP configuration described in T.3 is to execute a gate control list that will set the GateState to *open* for queue 6 and *closed* for queue 7 for a TimeInterval of T, and then set the GateState to *open* for queue 7 and *closed* for queue 6 for a TimeInterval of T, repeating ad infinitum. If there are no other traffic scheduling considerations, this can be achieved with a gate control list that contains just two SetGateStates gate operations, with parameters as follows:

1) GateState: 0, 1, 2, 3, 4, 5, 6 *open*, 7 *closed*, TimeInterval = T
2) GateState: 0, 1, 2, 3, 4, 5, 7 *open*, 6 *closed*, TimeInterval = T

This sequence of gate operations has the effect that during the initial time period T, the GateState for queue 7 is closed while queue 7 is being filled, and queue 6 is open to allow any queued frames to be transmitted; during the second time period T, the GateState for queue 6 is closed while queue 6 is being filled, and queue 7 is open to allow any queued frames to be transmitted. The gates for all other queues are open. The choice of time interval T is discussed in T.5; the cycle time (OperCycleTime; see 8.6.9.4.19) for the scheduled traffic state machines would be set to 2T in order to accommodate the sum of the time intervals for the two gate operations.

If there are traffic scheduling requirements for any of the other queues, then the gate control list could be extended to accommodate those requirements; however, the time interval between the changes of state of the gates for queues 6 and 7 has to be T, and consequently, OperCycleTime has to be a multiple of 2T, in order for ~~CQF~~sCQF requirements to be met. Figure T-2 illustrates the simplest possible traffic scheduling configuration for the case that traffic scheduling is only needed to support ~~CQF~~sCQF.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

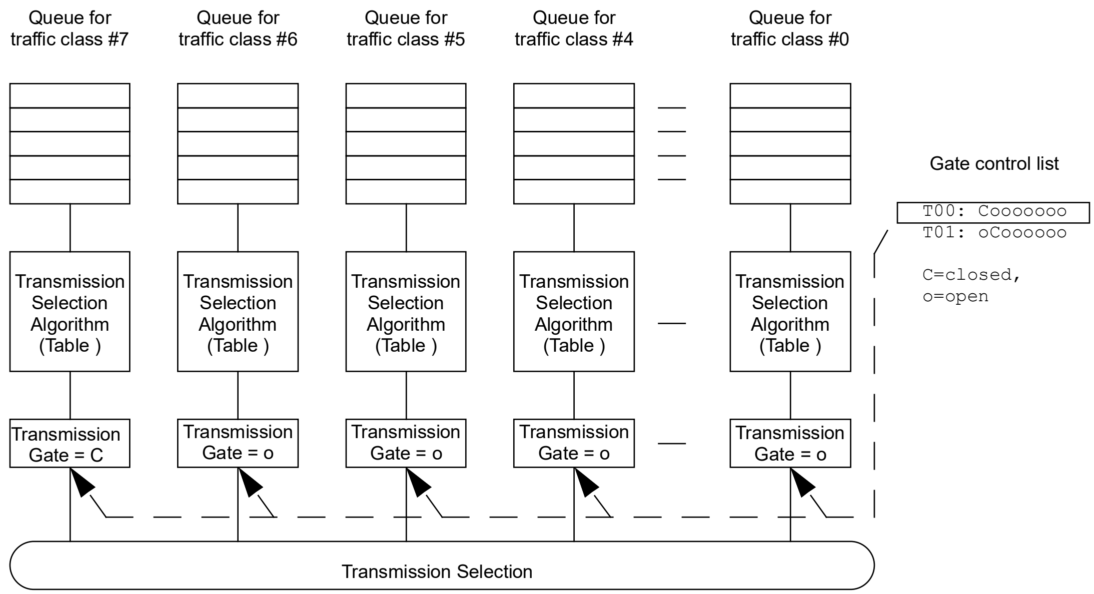


Figure T-2—Traffic scheduling example for ~~CQF~~sCQF

T.5 Timing considerations

T.5.1 Choice of T

T should be chosen such that it is large enough to accommodate the stream data that can be received during the class measurement interval for the stream(s) concerned, plus at least one maximal interfering frame or frame fragment. This is important in order to ensure the key performance aspect associated with the class measurement interval; namely, that if a stream or set of streams is observed over time, the reserved data rate for that stream or set of streams will not be exceeded during any observed time period equal to the class measurement interval associated with those streams. This effectively places a lower bound on the choice of T, that it should not be smaller than the class measurement interval, and also places a restriction on larger values of T, that they should be integer multiples of the class measurement interval.

If streams associated with two different observation intervals are being handled, for example if streams that use SR classes A and B pass through the Bridge, then the OperCycleTime used for the transmit traffic scheduling has to be a common multiple of the two class measurement intervals that are in use in order to make it possible for the transmission cycles to properly match the two values of T that are chosen. Figure T-3 and Figure T-4 illustrate how the Stream Filters, Stream Gates, and traffic scheduling could be configured in the case where SR classes A and B are active; in Figure T-3, incoming frames that carry SR Class A (priority 3) are handled using Gate 1, and the cycle time for the stream gate control list is twice the class measurement interval for SR Class A, which is 2 × 125 µs. Gate 1 alternately tags these frames with an IPV of 7 or 6. Incoming frames that carry SR Class B (priority 2) are handled using Gate 2, and the cycle time for the stream gate control list is twice the class measurement interval for SR Class B, which is 2 × 250 µs. Gate 2 alternately tags these frames with an IPV of 5 or 4.

The traffic schedule is based on the smaller of the two class measurement intervals, 125 µs, but now has four entries in the gate control list (as opposed to 2 entries in Figure T-2), giving an overall cycle time of 500 µs. The gate control list switches the gate states for traffic classes 7 and 6 every 125 µs, and switches the gate states for traffic classes 5 and 4 every 250 µs.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

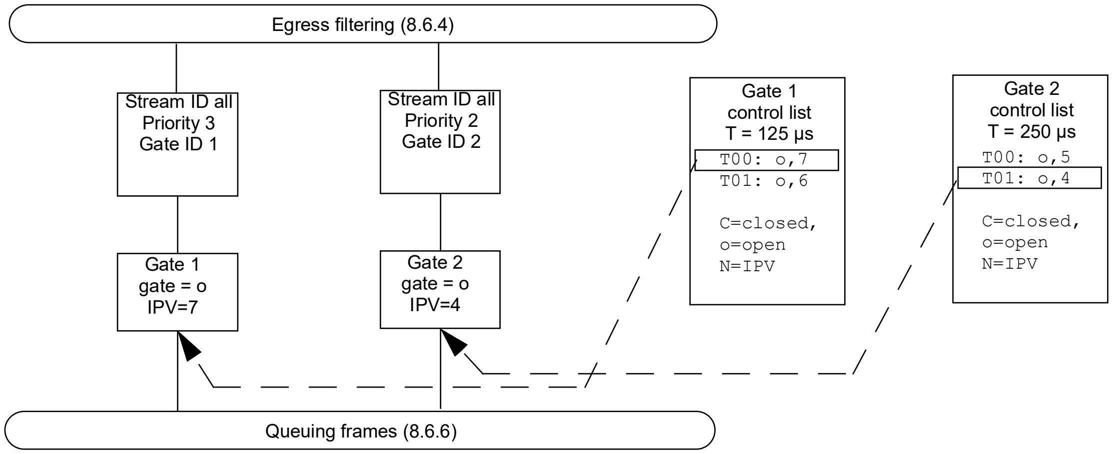


Figure T-3—Example Stream Filter and Stream Gate configuration with two values of T

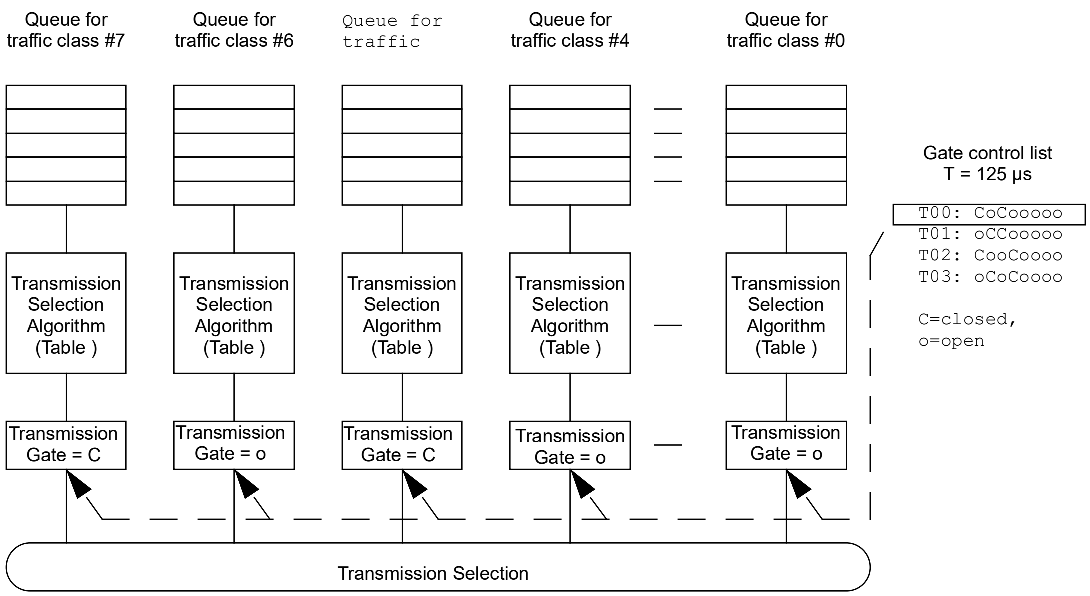


Figure T-4—Traffic scheduling example with two values of T

T.5.2 Cycle interleaving

In some circumstances, particularly where the data rates differ between reception and transmission Ports, it can be desirable to interleave cycles on the faster Port so that the best use is made of the higher bandwidth available, and also to reduce the latency that is added as a stream passes through faster parts of the transmission path. Because there is a delay imposed on the transmission of received frames, caused by the cyclic switching of reception and transmission between a pair of queues, when a queue is allowed to transmit, all of its received frames will have been enqueued, and all of them will therefore be transmitted in a burst, assuming that priorities permit, and that the transmission queue uses the strict priority transmission selection algorithm (8.6.8.1). Hence, if the received traffic from a given Port was spread out over the time interval T, and it is all sent to the same queue, the transmitted traffic will be compressed into a burst. If

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

the transmission data rate is, say, ten times the reception data rate, then the maximum length of that burst is T/10, so there is the potential to fit 9 more such bursts into the bandwidth available on that transmission Port. With appropriate timing on reception Ports and transmission Ports, the reception and transmission cycles can be interleaved such that those additional transmission bursts can occur. In the example illustrated in Figure T-5, it is assumed that:

a) There are two Ports on which stream data is being received, Rx1 and Rx2.
b) There is a single Port on which stream data is being transmitted, Tx1.
c) Rx1 and Rx2 operate at half of the data rate of Tx1 (or less).
d) All stream traffic is SR Class A, and is received with priority 3.

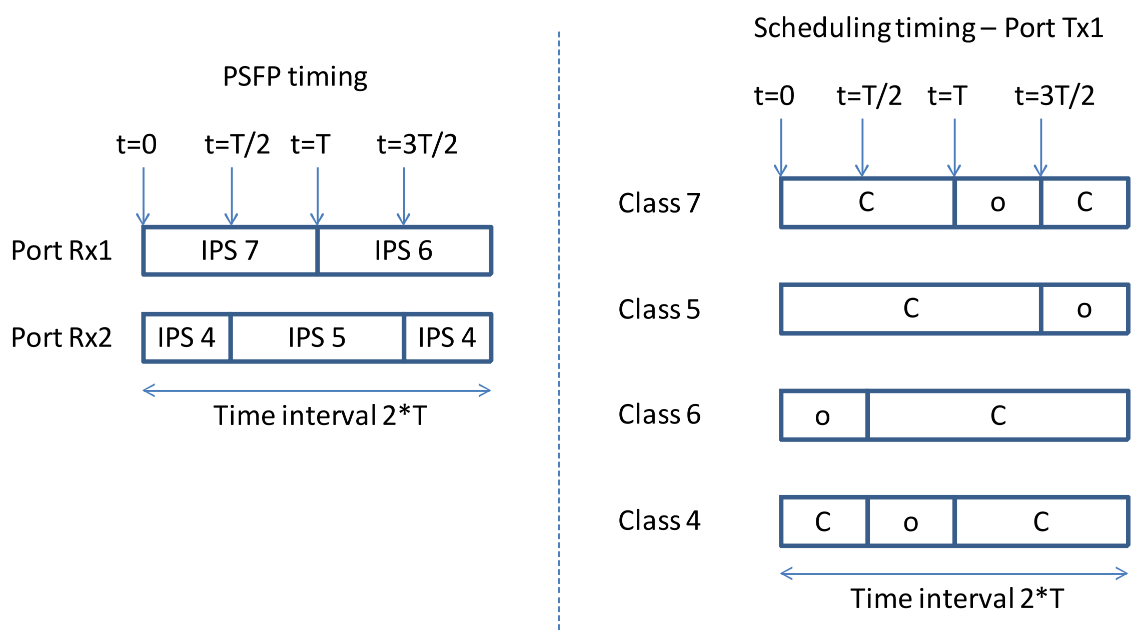


Figure T-5—Interleaving example—factor of 2

On the reception side, traffic received on Rx1 with priority 3 is sent to traffic class queue 7 during the odd cycles and to traffic class queue 6 during the even cycles. Similarly, traffic received on Rx2 with priority 3 is sent to traffic class queue 5 during the odd cycles and to traffic class queue 4 during the even cycles; however, these cycles are offset with respect to the cycles for Rx1 by T/2.

On the transmission side, each queue is in the open (transmitting) state for T/2, and in the closed (receiving) state for 3T/2. The timings of the gate open/gate closed events are arranged so that only one queue is transmitting at any one time.

It should be noted that with a scheme like this, although the transmit side appears to be operating on the basis of a value of T that is half that used on the Rx side, the effect from the point of view of the streams originating from a given reception Port is that the Rx and Tx timing is the same, as frames received on that Port are transmitted once in every time period T, and as assumed in the preceding point c), the transmit rate is twice the receive rate (or more), so the available transmit bandwidth is the same as or more than the receive bandwidth. The interleaving therefore meets the requirements stated in T.5.1. The reception Port of the Bridge downstream of Tx1 can operate using T/2, and if it is possible to carry this through to the transmission Ports as well, the contribution to the latency for streams passing through this Bridge will be T/2, rather than the T contributed by the first Bridge.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

More complex schemes can be envisaged; for example, using more than two Rx Ports, and these can be made to work as long as the received bandwidth can be shared equally between the pairs of traffic classes that are used. Interleaving of this kind can also be defined for larger interleave factors; the only limitation is the number of available outbound queues. It is also possible to define interleaving schemes where the received bandwidth is not shared equally among the pairs of traffic classes, as long as the bandwidth allocated to a given pair of traffic classes does not exceed the bandwidth available during the time intervals when those traffic classes are able to transmit.

NOTE—Although the number of traffic classes described in this standard is limited to 8, the value of the IPV does not have such a limitation placed upon it. If a system supported more than 8 traffic classes, it would therefore be possible to define interleaving factors greater than 4.

T.5.3 Cycle alignment between adjacent Ports

The examples so far assume a perfect world where the transmission from transmission Port to reception Port is instantaneous and the internal timings of the transmitting and receiving systems are perfectly synchronized. In reality, transmission takes time, and synchronization is not perfect; therefore, it would be possible that a transmission Port launches the last frame(s) of one transmission burst just after the reception Port downstream has switched reception and transmission queues, which would mean that those last frames are placed in the wrong queue. Similarly, if the timing misalignment worked the other way, it would be possible for the transmission Port to finish transmitting its burst early, and switch to transmitting the next burst before the downstream Port had changed state.

In order to avoid this problem, the timings must be adjusted such that there is a very high degree of probability that when a reception Port changes the state of the stream filters to direct incoming frames to a different outbound queue, there are no frames still to be transmitted, or in flight, from the upstream transmission Port. This can be achieved by slightly delaying the start of the transmission window, and slightly advancing the end of the transmission window. The value of “slightly” depends on a number of factors, including the following:

a) Any error in the time synchronization between the adjacent systems.
b) Jitter in the propagation time of a frame from starting to leave the transmit queue in the upstream system to being presented to the downstream policing function.
c) Jitter in the propagation time of a frame between the downstream policing function and the appropriate transmission queue in the downstream system.
d) The size of any potential interfering frame or frame fragment.
e) Difference in the resolution of the clocks that are maintained by adjacent systems.

The effect of this adjustment factor, S, on the timings shown on the transmission Port in the earlier examples would be that the time slots where the gate is in the “open” state would be shorter by a factor of 2S, and would start S later. Hence, the transmission phase for traffic class 7 in Figure T-5 would start at T+S, and would end at (3T/2)–S. S should be set to the sum of the errors or jitter values from all sources given in the preceding list.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Annex X

(informative)

Bibliography

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

[B1] Finn, Norman, https://mentor.ieee.org/802.1/dcn/21/1-21-0056-00-ICne-input-synchronization-for-cyclic-queueing-and-forwarding.pdf

[B2] Seaman, M., "Paternoster policing and scheduling" http://www.ieee802.org/1/files/public/docs2019/cr-seaman-paternoster-policing-scheduling-0519-v04.pdf,

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Annex Y

(informative)

Bin Cyclic Queuing and Forwarding

Insert a new Annex:

Y.1 Principles of BCQF

Y.1.1 Overview

Bin Cyclic queuing and forwarding (BCQF) is a method of transmission selection that can deliver deterministic, and easily calculated, latency for time-sensitive traffic streams. It is based on the following principles:

a) A Bridge egress queue using BCQF (a "BCQF queue") is notionally divided into bins. The bins are enabled for transmission serially, at a fixed interval T_C, which same (or nearly the same) value is used for some number of Bridges along the path of a Stream, said path constituting a BCQF segment of a network. At any given instant in time, a particular egress bin can be available for accepting frames for later transmission, or enabled for transmitting frames to the associated medium, or neither, but never both. See Y.1.2.

b) Each Stream utilizing a BCQF segment is allocated a certain number of bit times per transmission interval T_C. Steps are taken to ensure that no bin contains frames for any Stream that will take, in total, longer than that Stream's allocated bit times to transmit. Resource reservation ensures that the total bit times allocated over all Streams passing through a BCQF queue do not exceed T_C, even including possible interference from other queues on the port. See Y.1.3.

c) Frames assigned to the same bin at ingress to a BCQF Segment remain together in the same bin at each hop along the shared path. Two methods are provided to accomplish this, time-based bin assignment (tCQF) and count-based bin assignment (cCQF).

Taken together, these principles mean that no frames conforming to a Stream's bit time allocation are dropped due to congestion, and that the end-to-end delivery delay varies by little more than $\pm T_C$. End-to-end delay calculation largely reduces to a hop count (Y.3.9). These properties have significant consequences in larger networks, because they support the aggregation of Streams (Y.5), which can reduce end-to-end delivery times and/or reduce network resource requirements. Different queues on a single port can operate at different T_C values (Y.2) to provide BCQF facilities for different levels of latency and bandwidth requirements.Y.2

Y.1.2 BCQF transmission selection

A BCQF queue is described in this standard as being divided into bins, because this simplifies the procedures described for assigning frames to bins. In this formulation, each received frame is assigned an integer egress bin number *b* when queued for transmission on a port. This assignment can just as well be described in terms used for Asynchronous Transmission Selection (ATS) in §Clause 8. In ATS, each received frame is assigned a transmission time. If some number of frames are all assigned the same transmission time, selected from a range of future times separated by integral multiples of time T_C, this integer multiple is equivalent to the bin number *b*.

In previous versions of this standard, through IEEE Std 802.1Q-2022, each of the bins in the present standard were implemented using an entire class of service queue, and transmission gates were used to swap

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

between queues, thus rotating the bins. There was also a requirement that all of the Bridges in a network synchronize their transmission gates, and rotate the egress bins (queues) at the same time. This is a perfectly valid method for implementing BCQF. The present standard describes BCQF as a one or more individual class of service queues, each with multiple bins. This formulation offers a wider range of services.

BCQF class of service queues can be utilized on the same port with other transmission selection methods; strict priority determines which queue is selected for transmission.

Y.1.3 Bin selection

When a Stream frame is received and forwarded to a class of service egress queue that is enabled for BCQF, the frame is assigned to a particular bin in that queue. There are two methods for assigning a frame to a bin, time-based (tCQF, Y.3.1) and count-based (cCQF, Y.4). The same frame can be assigned to bins on two different egress ports using two different methods; the same bin can have frames from different Streams assigned to it using different methods. All of the frames in the same Stream received from the same port and transmitted on the same port use the same method. A Bridge can be configured for the bin selection method to use for all frames received from a given port, regardless of the egress port. It can be configured on an input-egress port pair basis. The selection method can be configured for specific Streams.

Y.2 BCQF in multiple queues on one egress port

Y.2.1 Multiple T_C model

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.

Instead of picking a single value for T_C that is sub-optimal for most Streams, we can apply multiple values of T_C to a single egress port, as shown in Figure Y-1.

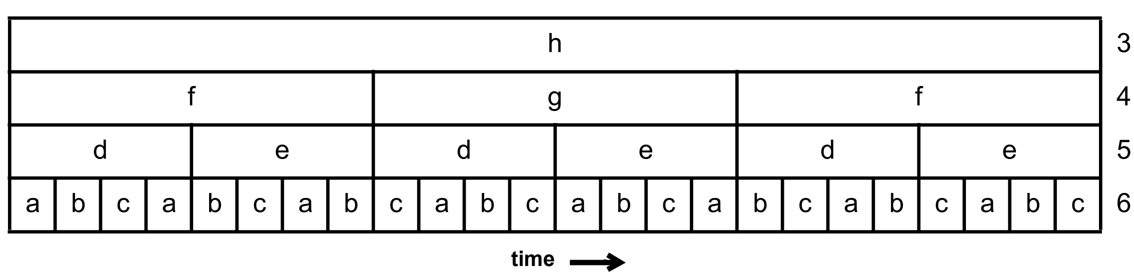


Figure Y-1—Multiple T_C values on multiple queues on one BCQF port

In Figure Y-1, we have a schematic timeline. Four class of service queues have been configured for BCQF, each with a different value of T_C. 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 Y-1 label which bin is transmitting 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; the others use two each.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

We assume here that the receiver of a frame can identify the particular BCQF instance (T_C value) to which the frame belongs by inspecting the frame. A TSN Bridge could use the 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 never 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 Y-2, where we illustrate the timing of transmission of frames from three levels of BCQF and the best-effort (BE) level. Note that BCQF 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.

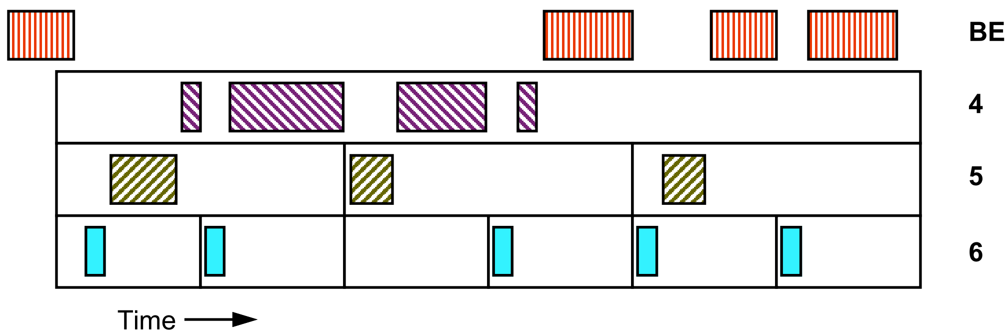


Figure Y-2—Transmission timing

For BCQF, 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) BCQF instance such that, at the Stream's bandwidth and frame size, the Stream occupies some space in every bin at that level. Then, BCQF 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 BCQF at 5 levels does not increase the bin memory requirements beyond that of 1-level BCQF. Allocating bandwidth to slow cycle times uses more buffer space, of course, because frames dwell for a longer time.

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

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

point of one frame per cycle, with no overprovisioning, and thus give a faster latency. Overprovision = higher 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.

Y.2.2 Integer multiples for T_C

The ideal would for each Stream S to have its own T_{CS} that requires no overprovisioning. 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 Y-3.

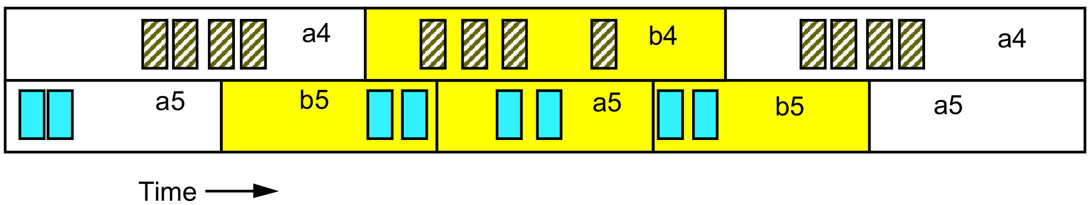


Figure Y-3—Variable T_C

In Figure Y-3, 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 Y.6.5), 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 transmit 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 transmitted during the period that b4 is transmitting. b4 cannot transmit 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 exactly into one cycle at the next-slower layer ensures that the lower-priority, slower cycle, will always have sufficient time to transmit all of its frame, because the problem in Figure Y-3 is avoided. Integral multiples fitting exactly means that, at the moment a cycle starts and ends at one priority level, a cycle starts and ends at each higher priority level, as illustrated in Figure Y-1. This scheme also bounds the number of preemption events that can steal bandwidth from a given priority level (see Y.3.3).

Y.2.3 Admission control for multiple T_C values

Y.2.1 describes the operation of BCQF with multiple T_C values operating simultaneously on one egress port. Figure Y-2 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 BCQF 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 BCQF priority level reduces the time allocable to other priority levels; there is only one physical link.

An administrator may wish to restrict allocation of BCQF transmission times to leave room for transmitting non-BCQF 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 BCQF levels on all of the egress ports through which the Stream travels have not been exhausted. At any given

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

BCQF priority level x, one can add the bits allocated to all Streams in one cycle at BCQF priority level x, plus the sum over all more-important BCQF 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 BCQF priority level, there is the same percentage of dead time and slop for inaccuracies, but this is not necessarily the case.)

Y.2.4 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 BCQF priority levels, and initiate all transmissions in that order, at line rate, without introducing extra inter-frame gap time. Since, with BCQF, no bin has frames both arriving and being transmitting at the same instant, this should pose no insurmountable problems for implementors.

Y.3 Time-based Cyclic Queuing and Forwarding (tCQF)

Y.3.1 Frequency lock requirement

tCQF does not require synchronization of the system clocks, but does require frequency lock. That is, the number of BCQF cycles in two Bridges that are frequency locked must be the same, over an arbitrarily long interval of time.

Y.3.2 Timeline for time-based bin assignment

We have two Bridges, A and B. Both are running BCQF on each of multiple ports. The receiver is using tCQF to assign frames to bins.

When a BCQF cycle starts on a particular port, Bridge A transmits all of the frames in one bin towards receiving Bridge B, not necessarily in a single burst. After some gap following the transmission of the last frame in the bin, and at time T_C after the cycle started, another cycle starts. At this point, it starts transmitting the frames from the next bin. The cycle in both Bridges happen regularly, with the same period T_C. At the next hop, Bridge B must be able to assign each received frame to a transmit bin such that 1) frames that were in the same bin in Bridge A, and are transmitted on the same port from Bridge B, are placed into the same bin in Bridge B; and 2) frames in different bins in Bridge A are placed in different bins in Bridge B.

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

T_C	nominal (intended) period of the BCQF cycle
T_I	maximum interference from lower-priority queues, either 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 Bridge A by Bridge 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 egress from B

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

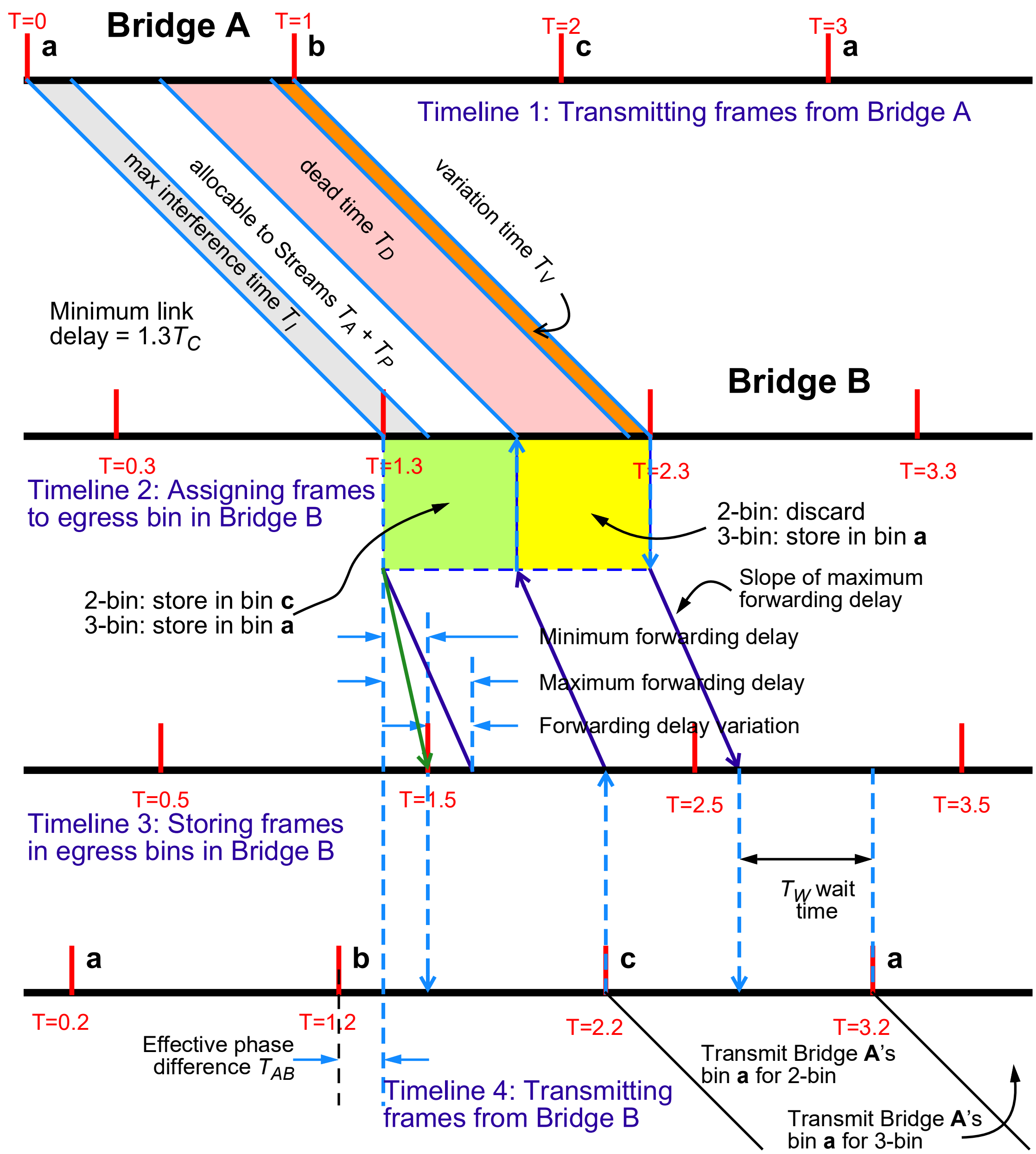


Figure Y-4—Example of timelines for time-based CQF

For time-based CQF, T_{AB} must remain constant; that is, any variation in T_{AB} 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 Y-4 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 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 egress 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 cycle start times in anticipation of the time

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

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 the start of a cycle.

Y.3.2.1 Egress timeline 1

Figure Y-4 shows an interference delay T_I (the gray area) between the start of Bridge A's cycle (the red ticks in Figure Y-4) 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 Y-4 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 BCQF Stream. This value is always bounded. See Y.3.3.

There can be some variation in the time from the selection of a frame for transmission in Bridge 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 Bridges' clocks caused by accumulated frequency variations, asymmetrical links, etc., causes uncertainty between the transmitting and receiving Bridges' 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 BCQF cycles 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 Y.3.5, 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.

The total time per cycle that can be used for transmitting Streams is, then:

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

This T_A is a maximum, local to a particular class of service and egress port on a Bridge. 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 start of the dead time. 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 Bridge to which the egress port is connected. T_V can be supplied either by the time sync implementation, by configuration, by summing the contributions of Bridge A and Bridge 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.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

The last frame transmitted from the bin has to complete transmission within the period marked T_A in Figure Y-4. (But, see Y.6.6.)

Y.3.2.2 Receive timeline 2

The timeline at the receiving port is timeline 2 in Figure Y-4. 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, this standard assumes that each frame is assigned to a bin on an egress port based on the timestamp (Clause 90 of IEEE Std 802.3-2018) on the frame. Other means of assigning an arrival time to a frame can be used.

A critical aspect of timeline 2 is its offset from timeline 4, the egress timeline. This offset is shown as T_{AB} in Figure Y-4. 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 Bridges A and B, and 2) measuring the link delay from Bridge A to Bridge 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 Y-4 can be computed. The phasing of the Bridges' egress 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, because cycle start times are not necessarily synchronized among the Bridges in a network. One could even adjust the phasing (by adjusting the phase of timeline 4) to favor certain paths through the network.

For time-based CQF, if a frame (belonging to a Stream) is received that straddles a cycle (first bit in one cycle on timeline 2 of Figure Y-4, 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 Bridge 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.

Y.3.2.3 Storing frames timeline 3

The timeline at the point where frames are stored into an egress bin is timeline 3 in Figure Y-4. The red ticks on timeline 3 mark the earliest point at which the first frame transmitted from a particular bin could reach the egress 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 egress queue. The maximum forwarding delay is also shown. The forwarding delays shown in Figure Y-4 include the time to install the frame in the egress bin and for its presence to filter through to the point that it can be selected for transmission.

For a Bridge B that is connected to and receiving BCQF frames from n other Bridges, we have n bin assignment problems to solve, one for each input port on Bridge B. The problem for each is to determine how many bins are needed, and to which bin each frame is to be assigned.

There are two bin assignment methods shown in Figure Y-4: the 2-bin method, in which the frames received from Bridge A bin a are assigned to bin c in Bridge B, and the 3-bin method, where those same frames are assigned to bin a in Bridge B. The slope of the maximum forwarding delay allows us to compute the latest moment at which frames received from bin a on Bridge A can be stored into bin c on Bridge B. The shaded areas just below timeline 2 in Figure Y-4 show the time windows for bin assignment. If two egress bins are used, then frames received from bin a on Bridge 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 Bridge 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.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Time T_W in Figure Y-4 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 Y-4.

Unlike timeline 1, timeline 2, or timeline 3, the red ticks on timeline 3 are not hard boundaries. The forwarding delay variation shown in Figure Y-4 could, in theory, be longer than one cycle time T_C. See Y.3.6.

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 (Y.2.4) can be met.

Y.3.2.4 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 Y-4 shows the two options for the choice of which egress cycle in Bridge B is used to transmit frames that were transmitted from bin a in Bridge A.

Y.3.3 Preemption and interference

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 Y.3.1 and Figure Y-4) on the allocable bandwidth. Frame preemption is described in §6.7.2 and in Clause 99 of IEEE Std 802.3-2018. Preemption can reduce the interference time.

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 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 Y-4.

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 Y-1, 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.

Y.3.4 T_C computation

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

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

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

As described in Annex T, scheduled CQF (sCQF) 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 sCQF.
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-transmission 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.

Y.3.5 Calculation of dead time T_D

Timeline 3 in Figure Y-4 shows the calculation of T_D, which applies only to 2-bin mode. The starting point of T_D is the moment that the egress 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 Y-4. 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 Bridge 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 Bridge A so that Bridge 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 Bridge, either via management or via an extension of the reservation protocol.

<< Editor's note: No such mechanism exists, at this point. >>

b) A Bridge 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 a previously-computed Stream's end-to-end latency.
c) All Bridges in a network can be configured with a reasonable maximum value for T_D. If a particular input/egress port pair on a particular Bridge computes a value for T_D that exceeds this maximum, then 3-bin operation is required.

Y.3.6 More than 3 egress bins

So far, the discussion of Figure Y-4 assumes that the variation in forwarding delay is small, relative to T_C. If this is not the case, Bridge B can use more than 3 egress bins, and assign received frames to bins whose transmission is scheduled far enough ahead in time to ensure that, in the worst case, they will arrive in the proper 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 egress port.

In certain situations, e.g. when a Stream is replicated 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, extra egress bins can be allocated, and used to impose a delay of an arbitrary number of cycle times T_C on every frame.

Each egress port in a Bridge, and each egress 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 egress port, while another

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Stream, which needs a path-matching delay, can use 12 bins on the same port. (Of course, this requires per-Stream configuration.)

Y.3.7 Deterministic behavior of time-based CQF

tCQF guarantees the Deterministic QoS by the following argument.

We assume that the Talker uses BCQF. Non-BCQF inputs to a Bridge are discussed in Y.4.4.

We consider only one value of T_C along the path of a given Stream from Talker to Listener. Y.2.3 and Y.3.8 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 Y.3.1 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 tCQF 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 BCQF egress bin after entry to the network. That is, the frames from a given Stream do not exceed the Talker contract in the first BCQF egress bin in the network.

   Ingress conditioning and/or policing is discussed in Y.4.4.

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

   Y.3.1, and particularly Figure Y-4, 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 egress cycle, then cycle integrity is maintained. (Of course, this only works for point-to-point links.)

c) There is no fan-in for a single 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 egress 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 egress port's available transmit time. The implications of such a limitation are obvious.)

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Y.3.8 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 egress port with cycle time n*T_C, then n successive input cycles can be deposited in the same egress 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 egress 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 CQF step (see Y.4.4), to ensure that the Stream never exceeds its contract in the new cycle time.

Y.3.9 Computing the actual end-to-end latency for time-based CQF

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 Y.3.1. Referring to Figure Y-4, 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 egress window in which a frame received in that input window will be transmitted. This is the dwell time for the frame in this Bridge. Maximum and minimum times for this delay are given in 100.1.2.

Link delay is relevant to the computation of end-to-end delay, but it can be hidden by using time-based CQF in time-synchronized Bridges, and using dead time, so that the link delay is accounted for within the BCQF cycle time T_C. If the link delay does need to be added to the delay, it is the one-way link delay that is added. Typically, this is measured using the PTP. 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 Y-4, we can see that the difference between using two and three bins for a given input-egress 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 egress 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.

Y.3.10 Egress bin selection

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

We describe here one method for receiving a frame and assigning it to a bin. There are many ways to accomplish the same task.

Let B_o be the number of physical egress 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.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

At the egress 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 egress cycle.

When a frame arrives at an egress 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 egress port o and B_n is the number of egress bins on the port. See Y.3.11.4 for the determination of P_{io}. Note that in the extreme case of all egress ports using two bins, all synchronized, and all input cycles in phase with the egress cycles, the table P_{io} reduces to a single value, 0 or 1.

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 BCQF service guarantees.

Y.3.11 Parameterization of time-based CQF

Let us go through the exercise of initializing an input/egress port pair for tCQF. 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 tCQF.

Y.3.11.1 Cycle wander

Adjacent Bridges must be frequency locked as described in Y.3.1. For any given port, there is a worst-case system clock difference 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 the difference 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.

We assume that bin rotation operate under control of a clock that is local to a port. The management controls that configure the rotation are defined in terms of a system clock. The Bridge can align 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. This feeds into the calculation of T_A in Y.3.2.1.

Y.3.11.2 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.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Y.3.11.3 Calculating the number of bins required

The procedure to calculate the number of bins needed on an egress 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 egress 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 egress 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 egress 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 egress 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 egress 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 egress 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/egress 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 egress port’s cycle; and/or
— Imposing implementation-specific limitations on the Streams, e.g. reducing fan-in to an egress 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.

Y.3.11.4 Initial bin phase

The number of bins required on an egress port is the maximum required over all input ports. This may be further increased by intentional delays (Y.3.6). When initializing an input port, a correspondence must be

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

made between the input and egress ports, so that a frame received on the input port will be stored in a particular bin in the egress port, the one that will become the transmitting bin in the appropriate number of egress cycles in the future.

The phasing between input and egress ports' cycles, and thus the number of bins in port *o* used by port *i*, is determined by the P_{io} table defined in Y.3.10. We compute P_{io} when initializing BCQF, or when the relative phase of the input and egress 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 egress port *o* at time T, B_{io} is the total number of bins required of egress 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 egress ports.

Y.3.11.5 Dead time / bandwidth balance calculation

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 egress 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 Y.3.11.3. There are, at the very least, the following ways to make this decision:

— Configure the egress 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 egress cycle phase on all Bridges. Configure minimum and maximum allocable bandwidth values for each BCQF 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 egress port cycle phasing to optimize the delay for certain specific Streams.

Y.4 Count-based Cyclic Queuing and Forwarding (cCQF)

As described in Y.3.1, time-based bin assignment assigns frames to egress bins based on the time of arrival of the frame, and requires that the egress queues of successive hops along a cCQF 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 egress port. Then, the egress queues along the path can run nearly the same frequency, and their relative phases (T_{AB} in Figure Y-4) can diverge.

cCQF is an alternative description of the paternoster algorithm defined in [B2]. It 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 BCQF 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.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Y.4.1 Calculating allocable time T_A

Count-based CQF computes the length of the portion of a cycle that can be allocated to Stream data, T_A, in a manner similar to that used for time-based CQF in Y.3.2.1 and Figure Y-4:

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

Again, T_C is the nominal cycle length, T_I is the interference from lower layers (Y.3.2.1, Y.3.3), T_P is the penalty incurred if this priority level is preemptable (Y.3.3), is the T_D is the dead time imposed by Bridge to which this Bridge is transmitting (Y.3.2.1). However, the time-based calculation uses T_V for the last time, a catch-all for discrepencies including output delay, link delay, clock accuracy, and timestamp accuracy. The count-based calculation uses:

T_X worst-case difference between the receiver's actual T_C values and the T_C value by which the Talker's reservation is defined.

All of the items included in T_V in Y.3.2.1 are irrelevant to count-based CQF:

a) Output delay, link delay, and clock accuracy affect only the phase relationship between the transmitter's cycle the receiver's egress ports' cycles. (In time-based CQF, there is no long-term clock error.)
b) Timestamps are not relevant to count-based CQF.

Persistent differences in T_C, however, are important. If the transmitter's cycle time is shorter than the a receiver cycle time, and if the Talker is generated data over the long term that keeps every transmitter cycle full to the limit of a Stream's reservation, then the receiver would eventually have to drop frames. The term T_X ensures that each count-based CQF hop can serve the Streams allocated to it. If a transmitting port is faster than the receiving port, and thus builds up a extra frame in the receiver's bin(s), then in the long term, even if the Talker runs continuously, the transmitter will eventually run ahead of the Talker, and have a less-than-full cycle. This gives the net-hop receiver a chance to catch up.

Y.4.2 Dead time T_D

Count-based CQF in a receiving Bridge cannot impose dead time (Y.3.5) on the transmitting Bridge; it has no need to. However, it may have dead time imposed upon it if it transmits to Bridge using time-based CQF.

Y.4.3 Number of egress bins

In an ideal world, only two bins are required per egress queue for count-based CQF, one filling and one transmitting, The fact that successive Bridges employing count-based CQF have slightly different actual values for T_C makes a third bin necessary, because all of the frames destined for one bin (the one that is filling) do not necessarily all arrive during the time when the filling bin is open. If the forwarding delay variation shown in Figure Y-4 is non-0, as it is in most implementations, at least one more bin, the fourth, is necessary. Additional bins can be added to accommodate input Streams from devices that are not transmitting using BCQF. Assuming that such input Streams can be characterized by a committed burst size (§8.6.5.5), this committed burst size can be added to the basic three bins to calculate the total number of bins required.

Y.4.4 Using both count-based and time-based CQF

A Stream entering a Bridge from a correctly-configured Bridge or end station that runs BCQF, and that has reserved bin space allocated for it, will not disrupt the deterministic behavior of BCQF. However, a BCQF Bridge could receive input from a Bridge, a Talker, a router, or any other device that uses some deterministic

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

algorithm(s) to condition its Streams, but uses an algorithm other than BCQF. We assume that reservations (contracts) for these Streams can be translated into BCQF terms, with perhaps some overprovisioning required. In the long term, the Talker adheres to the contract, and will not disrupt determinism. But, since the transmitter is not using BCQF, we cannot use just a frame's arrival time to assign it to a bin, except by overprovisioning BCQF 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 transmission methods described for time-based bin assignment in Y.3.1, but instead of obtaining the bin selector S from the time of receipt of the input frame, as described in Y.3.10, it uses a state machine dedicated to each BCQF 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 CQF can be used.

A given egress 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.

Y.5 Stream Aggregation

Stream aggregation is useful for both scaling up the number of Streams that a network can support, and for decreasing the end-to-end latency of Streams that are aggregated. 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. Stream aggregation can work whether the frames are encapsulated in some common wrapper, or whether they are simply treated identically (e.g. all given the same IEEE Std 802.1CB stream_identifier).

This standard does not specify any protocol for encapsulating aggregated Streams.

The aggregated Stream has a single reservation that is the union its component Streams' reservations (Y.6.1). Ideally, his higher-bandwidth Stream can be assigned a BCQF priority level with a faster T_C than its components can use. For example, instead forwarding 10 Streams, each with one frame, in a 1 millisecond T_C bin, a Bridge could be forwarding one aggregated Stream that has one frame in a 100 microsecond bin, Buffer space requirements are cut by 90%, per-hop delay by up to 90%, and state machines, e.g. count-based bin assignment machines, by 90%.

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.

Figure Y-5 illustrates the value and the limitations of Stream aggregation. In this figure, there are three Streams, all entering Bridge A, and all three traversing the same path at least as far as Bridge E. Bridge A operates on three separate Streams in the usual manner for BCQF, placing one frame from each stream into each egress bin. In Bridge B, the three Streams are aggregated into a single aggregated Stream. This Stream

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

is forwarded through Bridges C and D. Bridge E dissolves the aggregated Stream, distributing the component Streams' frames on three ports.

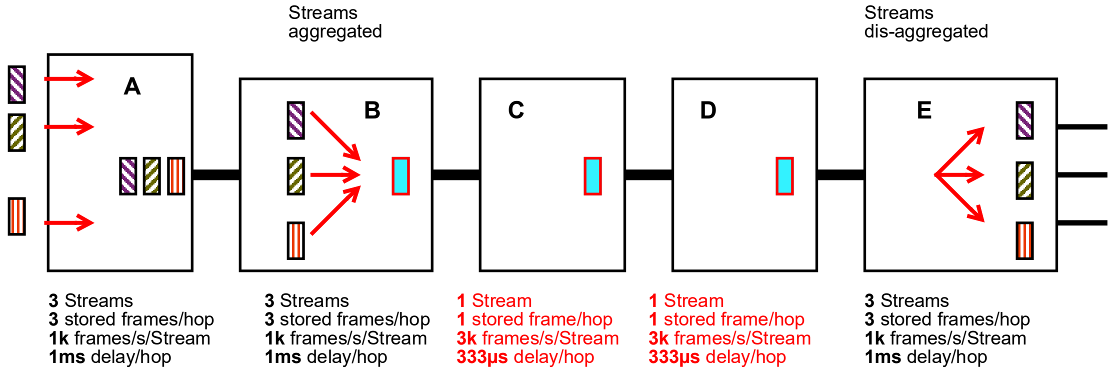


Figure Y-5—BCQF Stream Aggregation example

As indicated in the captions in Figure Y-5, the aggregating Bridge B requires the same amount of buffer space and state machjines, and imposes the same forwarding delay on all three Streams, as does Bridge A (see Y.5.1). Once aggregated, however, Bridge C and Bridge D forward the Streams with less delay, and using less buffer space and fewer state machines, than Bridges A and B. Finally, Bridge E dissolves the aggregated Stream into its components (see Y.5.2). Bridge E, again requires the same resources imposes the same delay on the three Streams as the non-aggregated Bridge A. The case of Stream mixing, that is, disaggregating Streams and then re-aggregating them in different combinations in the same Bridge, is described in Y.5.4.

Y.5.1 BCQF Stream aggregation

Stream aggregation usually includes changing the cycle time from a slower to a faster cycle. Changing the cycle time is discussed in Y.3.8. In general, count-based bin assignment is required, because the Streams being aggregated can be arriving from different ports, and be distributed along the duration of a large cycle time. On the port transmitting the aggregated Stream, the bins are typically rotated at a higher rate, and often, are on a medium with higher bandwidth than the media on which the Streams arrived. Typically, the input Streams' frames can arrive in lumps, with intervening gaps, due to fan-in (multiple input ports to a single egress port). In the worst case, the same number of egress bins are required on the egress port as would have been needed had there been no aggregation.

Y.5.2 BCQF Stream disaggregation

Stream disaggregation usually includes changing the cycle time from a faster to a slower cycle. Changing the cycle time is discussed in Y.3.8..

Whether count-based or time-based bin assignment can be used in the disaggregating bridge depends on the degree to which the Bridges along the path of the aggregated Stream are able to maintain the timing of the aggregating Bridge's inputs. In theory, if time-based CQF is used all along the aggregates Stream's path, and if the disaggregating Bridge can align its input cycles with the aggregating Bridge's input cycles, in spite of the intervening Bridges, then it would be possible to use time to select the egress buffers in the disaggregating Bridge. No mechanism is provided in this standard to achieve that synchrony.

Assuming that the disaggregating Bridge uses count-based bin assignment, the frames of the individual disaggregated Streams are distributed into the slow egress bins in the normal manner.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

The number of bins required in the disaggregating Bridge may be slightly larger than the number of bins required for normal forwarding of the equivalent disaggregated Streams, because the Bridges carrying the aggregated Stream (e.g. Bridges C and D in Figure Y-5) can introduce a small amount of delay variation. Any such variation increases the buffer requirements in the disaggregating Bridge. See Y.5.3.

Y.5.3 BCQF delay variation and disaggregation buffers

One of the major issues with aggregating Streams is the amount of buffer space required at the disaggregation point. For example, suppose that Bridges B, C, and D in Figure Y-5 use Asynchronous Traffic Shaping, instead of BCQF, to forward the aggregated Stream. Because the aggregated Stream is allocated essentially the sum of the bandwidths of its component Streams, if some Streams in the aggregation are flowing intermittently, then other Streams can be given the bandwidth not used by their associates in the aggregation. This can easily result in bursts and gaps in the individual Streams when finally delivered to the disaggregation Bridge (Y.5.2) or mixing Bridge (Y.5.4).

Count-based CQF makes a critical reduction in the delay variation of the aggregated Stream, even when components of the stream are intermittent or missing, as long as the number of bins in each hop is minimized. Time-based CQF adds no delay variation; the aggregating Bridge's bin assignments are maintained all the way to the disaggregating Bridge. The disaggregating Bridge performs count-based bin assignment, but requires excess buffer space only in amount equal to the size of the input bins on the aggregating Bridge (Bridge A in Figure Y-5), as its egress cycles are not necessarily in phase with the aggregating Bridge's input cycles.

Y.5.4 BCQF Stream mixing

Stream mixing occurs when Streams are disaggregated and then reaggregated, perhaps in different combinations, in the same Bridge. If we neglect the actual operations of wrapping and unwrapping the frames (if needed), this operation is very similar to either aggregation or disaggregation; mixing does *not* require a set of disaggregation buffers followed by a set of aggregation buffers.

In general, any Stream that does not pass intact through a Bridge has to be split into its component Streams. (Split at the topmost level—aggregations of aggregations need not be split all the way down the stack). Each component Stream passes through a count-based bin assignment state machine, and is assigned an egress bin appropriate for its egress port, whether nor not it is also being reaggregated. As for the case for disaggregation (Y.5.2) the egress buffer space required depends on the cycle times of the original, component Streams at the time they entered the aggregation.

Y.6 Additional considerations

Y.6.1 Computing the BCQF reservation

At the lowest level, e.g. in a count-base bin assignment state machine, BCQF reservations are in terms of bit times per cycle, with an implied cycle time. However, Streams are not usually characterized in this manner.

<< Editor's note: This section is clearly incomplete. It will include a description of how to convert TSN and CBS specs to BCQF, and also how to combine multiple Streams' TSN and/or CBS specs into a single BCQF spec. >>

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

Y.6.2 Frame size problem

Stream data does not always consist of frames that are all the same size. 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 + 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 best-effort traffic, so overprovisioning may not be a serious concern.

Another approach to increased resource utilization efficiency is to run each BCQF Stream, on ingress to the TSN network, 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[5] into large transmission units.)

Y.6.3 Tailored bandwidth offerings

We can note that, in a service provider environment, overprovisioning can be almost eliminated by a combination of 1) Stream Aggregation (Y.5) 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.)

Y.6.4 Overprovisioning to improve latency

A minimum of network resources is consumed when a Stream with constant frame size is allocated just enough bit times per cycle for a single frame, and the cycle time is 1/(the Stream's frame rate). One frame is delivered per cycle.

[5]Not to be confused with Link Aggregation

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

If that same number of bit times is reserved for that Stream in a class of service with a T_C that is, for example, five times shorter than the minimum-resource T_C, then that Stream will experience shorter end-to-end delay through the network. This lower delay comes at a cost; four out of five cycles have unused allocable transmission time. That bandwidth is available to best-effort traffic, but not to other Streams with reserved resources.

In a network that is not saturated with Stream traffic, this can be a viable trade-off.

Y.6.5 BCQF and credit-based shaper

Looking at Figure Y-2, we see that, once the major cycle at priority level 4 begins transmitting, the best-effort traffic is interrupted until all of the BCQF level 4 data is transmitted. At some point, as the amount of traffic in a very slow BCQF 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 BCQF cycle(s). However, the parameters of this shaper must be adjusted as the load on the slow BCQF 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 a significant advantage of time-based CQF—its freedom from requiring reconfiguring a Bridge each time a Stream is added.

Y.6.6 Interactions among BCQF, ATS, and control traffic

For time-based CQF to function correctly—in particular, for it to guarantee no congestion loss—all of the frames in a bin have to be transmitted before the beginning of the cycle. The formulas for buffer allocation and end-to-end delay depend on this. Therefor, to configure queues with ATS shapers at priority levers more important than a BCQF queue, one must be sure that, in the worst case, the BCQF queue can still empty its bin within the alloted period (T_A in Y.3.2.1).

These control protocols can include both link-local protocols, such as LLDP or BPDUs (<insert reference>), and protocols likely to be forwarded as data by a Bridge, such as Layer 3 routing protocols.

Similarly, steps have to be taken by the implementer and/or network manager to understand high-priority control traffic such as bridging or routing protocols. Typically, this means creating one or more Stream reservations to control the impact of control protocols on Stream data. Of course, the impact of the other Streams on the control protocols also has to be analyzed. Experience with TSN tends to show that such protocols, not having been regulated carefully before the advent of deterministic networking, are quite tolerant of the levels of delay imposed by transforming them from conflicting, highest-priority frames, to them reasonably high-priority, but bandwidth limited, Streams.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards draft, subject to change.

(informative)

Commentary

<< Editor's Note: This is a temporary Annex intended to record issues and their resolutions as the project proceeds. It will be removed prior to Standards Association ballot. >>

ZZ.1 Terminology

Comment #24 on D0.3 had the following resolution:

> We will use the term "CQF" for all versions, old and new. We will use the general term "CQF" as opposed to subdividing the term as much as possible.
>
> The term "Scheduled CQF" will be used when it is required to refer to the "old" CQF described in 802.1Q-2022 Annex T.
>
> When necessary, we use the term "Time CQF" for time-based CQF and "Count CQF" for count-based CQF.

The editor found that using "Time CQF" as the name of a set to which "time-based CQF" is one member was very confusing. Instead, the term "Bin CQF" has been used in this draft in place of "Time CQF". Thus, we have:

a) CQF, which covers all forms of Cyclic Queuing and Forwarding, which is divided into two classes, Scheduled CQF and Bin CQF.
b) Scheduled CQF (sCQF), which is the CQF from Annex T of IEEE Std 802.1Q-2018. It uses multiple class-of-service queues for each CQF priority level.
c) Bin CQF (BCQF), which is introduced in the present amendment, which has two forms, Time-based CQF and count-based CQF.
d) Time-based CQF (tCQF) assigns frames to bins based on arrival time.
e) Count-based CQF (cCQF) assigns frames to bins based on per-Stream state machines that count bit times on the output port.

The editor anticipates that this will not be the last word on terminology. For example, if we restore the old CQF to being CQF and call the new stuff something very different (e.g. TDM, tTDM, cTDM), then all of the changes to Annex A, Annex B, and most of the changes to Annex T disappear.

ZZ.2 Items left to do before first working group ballot

a) Comment #1 on D0.3: YANG modules needed. (This can be done after the Managed Objects have been defined.)
b) Comment #2 on D0.3: MIV modules needed. (This can be done after the Managed Objects have been defined.)

ZZ.3 Counting frames, not bits

Comment #18 on D0.3 has not been addressed:

COMMENT:

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

The usage of frame numbers in stead of bits would also be possible for some use-cases.

SUGGESTED REMEDY:

Discuss if count-base could also mean number of frames instead of bytes/bits

DISPOSITION:

ACCEPT IN PRINCIPLE. Current T-Spec is in terms of maximum number of max-size frames per time interval. In that case, we could just count frames, instead of bits, for count-based CQF. There are cases where that could work (time interval == TC). There are cases where that doesn't work. Certaibnly there are cases where bit counts are necessary.

Maintain this question in Annex ZZ.

ZZ.4 Frame format for CPAP

Comment #4 on D0.3 has been addressed, though not exactly as planned:

COMMENT:

The usage of an EtherType sounds okay for me. The inclusion in 802.1 AS / 1588 frames would bind the solution to a time-sync protocol. One advantage of the CQF is the "free-running" mode without any strict time-synchronization, only frequency locked. An optimization can be reached if the phases are alligned in time. A combination with LLDP sound also okay me.

SUGGESTED REMEDY:

Check if there can be an 802.1 (Q?) EtherType for Control Protocols with an own sub-type for each protocol. PCAP would be one of the 802.1Q Bridge Protocols. Alternatively check if one of the special IEEE Bridge MAC Adresses could be used with an own sub-identifier for PCAP.

DISPOSITION:

ACCEPT IN PRINCIPLE. Use an existing bridge protocol. Identify a place that we can use for CPAP and for future, similar protocols, and document its use for this purpose. ECP is the likely candidate, but editor is free to choose for next draft.

Add new editor's note asking if the choice made will cause anyone any problems.

WHAT HAS BEEN DONE IN THIS DRAFT

ECP is not usable; It has a version field, but this field is ignored on receipt, so CPAPDUs would be mistaken for ECP DUs.

The old 802.1D Spanning Tree BPDUs are a possibility. The are encoded with a dedicated LLC value (inoti SNAP), followed by a 2-byte “protocol identification” field that has the value 0 for spanning tree and 1 for the Generic Attribute Registration Protocol. We could add a third value (2) for other protocols. However, using something other than an EtherType is distinctly out of fashion, these days. It is not clear that everyone’s hardware is capable of using a non-SNAP LLC value as an identifier for marking outgoing frames for timestamping.

IEEE 802.3 Slow Protocols are not an option; CPAP may be transmitted too often to be a Slow Protocol.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

Copyright © 2023 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.