P802.1Qdv/D0.3
July 12, 2023
(Amendment to IEEE Std 802.1Q™–2022 as amended by IEEE Std 802.1Qcz™–2022)

Draft Standard for Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment: Enhancements to Cyclic Queuing and Forwarding

Developed by the

LAN/MAN Standards Committee
of the
IEEE Computer Society

Unapproved draft

Prepared by the Security Task Group of IEEE 802.1

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 partipants in the IEEE Standards Association ballot process, and will be updated as convenient. New participants: Please read these cover pages, they contain information that should help you contribute effectively to this standards development project.

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

Editors' Foreword

Throughout this document any notes with a Cyan background (as in this paragraph) are temporary, inserted by the Editors for a variety of purposes. They will be removed prior to SA Ballot and are not part of the normative text. To avoid gratuitous changes to page numbering between final WG ballot and initial SA Ballot, the number of cover pages should remain unchanged, with pages intentionally left bank marked as such. The SA ballot cover pages may contain information for the stages of that ballot (including, if appropriate, cross-references to text changed in the course of SA balloting). The records of participants in the development of the standard will be added pre-publication.

This draft is a proposed amendment to an approved standard. All that it has to show are the proposed changes (including additions) to the standard that it amends. However experience has shown that the development of an amendment that includes the minimum amount of text needed to meet this goal is undesirable. First, such a minimal amendment hands the task of combining the amended text with the base standard not just to an editor rolling up the base text and outstanding amendments into a new edition, but also to everyone who wants to use the standard before that rolled up edition is available, which might be ten years in the future. Second, few if any reviewers have the time to mentally undertake that roll-up process when reviewing each successive draft. Much of the base text can remain out of sight and out of mind, with the consequence that a developed amendment may add material that does not take advantage of material already in the approved, duplicate that material, or even contradict it. If the changes consist of many small fragments, the result may prove barely readable when the merge is done. Accordingly this amendment may contain more of the base text than may appear strictly necessary. The eventual aim is to include sufficient text to make the context of the additions clear without repeated reference to the base text, thus making the intended use of the amendment easier. In early drafts more material can be included, with the aim of making sure that all the text that needs to be reviewed or appreciated when contributing to draft development is readily available to reviewers. There is a known drawback to including this additional text. Commenters tend to assume that any text shown can be amended. Only new text introduced by an ***Insert*** editing instruction can be freely changed. Where base text is included as part of a ***Change*** editing instruction, changes are restricted to those that are within the Scope of the project (refer to the PAR).

Participation in 802.1 standards development

All participants in the standardization activities of IEEE 802.1 should be aware of the Working Group Policies and Procedures, and the fact that they have obligations under the IEEE Patent Policy, the IEEE Standards Association (SA) Copyright Policy, and the IEEE SA Participation Policy. For information on these policies see 1.ieee802.org/rules/ and the slides presented at the beginning of each of our Working Group and Task Group meeting.

As part of our IEEE 802® process, the text of the PAR (Project Authorization Request) and CSD (Criteria for Standards Development) of each project is reviewed regularly to ensure their continued validity. The PAR is summarized in these cover pages and a links are provided to the full text of both PAR and CSD. A vote of "Approve" on this draft is also an affirmation that the PAR and CSD for this project are still valid.

Comments on this draft are encouraged. NOTE: All issues related to IEEE standards presentation style, formatting, spelling, etc. are routinely handled between the 802.1 Editor and the IEEE Staff Editors prior to publication, after balloting and the process of achieving agreement on the technical content of the standard is complete. Readers are urged to devote their valuable time and energy only to comments that materially affect either the technical content of the document or the clarity of that technical content. Comments should not simply state what is wrong, but also what might be done to fix the problem.

Full participation in the work of IEEE 802.1 requires attendance at IEEE 802 meetings. Information on 802.1 activities, working papers, and email distribution lists etc. can be found on the 802.1 Website:

http://ieee802.org/1/

Use of the email distribution list is not presently restricted to 802.1 members, and the working group has a policy of considering comments from all who are interested and willing to contribute to the development of the draft. Individuals not attending meetings have helped to identify sources of misunderstanding and ambiguity in past projects. The email lists exist primarily to allow the members of the working group to develop standards, and are not a general forum. All contributors to the work of 802.1 should familiarize themselves with the IEEE patent policy and anyone using the email distribution list will be assumed to have done so. Information can be found at http://standards.ieee.org/db/patents/

Comments on this draft may be sent to the 802.1 email exploder, to the Editor, or to the Chairs of the 802.1 Working Group and TSNTask Group.

Norm Finn
Editor, P802.1Qdv
Email: nfinn@nfinnconsulting.com

Janos Farkas
Chair, 802.1 TSN Task Group
Email:Janos.Farkas@ericsson.com

Glenn Parsons
Chair, 802.1 Working Group
+1 514-379-9037
Email: glenn.parsons@ericsson.com

NOTE: Comments whose distribution is restricted in any way cannot be considered, and may not be acknowledged.

All participants in IEEE standards development have responsibilities under the IEEE patent policy and should familiarize themselves with that policy, see http://standards.ieee.org/about/sasb/patcom/materials.html

As part of our IEEE 802 process, the text of the PAR and CSD (Criteria for Standards Development, formerly referred to as the 5 Criteria or 5C's) is reviewed on a regular basis in order to ensure their continued validity. A vote of “Approve” on this draft is also an affirmation by the balloter that the PAR is still valid.

Draft development

During the early stages of draft development, 802.1 editors have a responsibility to attempt to craft technically coherent drafts from the resolutions of ballot comments and from the other discussions that take place in the working group meetings. Preparation of drafts often exposes inconsistencies in editor’s instructions or exposes the need to make choices between approaches that were not fully apparent in the meeting. Choices and requests by the editors’ for contributions on specific issues will be found in the editors’ Introduction to the current draft and at appropriate points in the draft.

The ballot comments received on each draft, and the editors’ proposed and final disposition of comments on working group drafts, are part of the audit trail of the development of the standard and are available, along with all the revisions of the draft on the 802.1 website (for address see above).

During the early stages of draft development the proposed text can be moved around a great deal, and even minor rearrangement can lead to a lot of ‘change’, not all of which is noteworthy from the point of the reviewer, so the use of automatic change bars is not very effective. In early drafts change bars may be omitted or applied manually, with a view to drawing the readers attention to the most significant areas of change. Readers interested in viewing every change are encouraged to use Adobe Acrobat to compare the document with their selected prior draft. Note that the FrameMaker change bar feature is useless when it comes to indicating changes to Figures.

Project Authorization Request, Scope, Purpose, and Criteria for Standards Development (CSD)

The complete PAR, as approved by IEEE NesCom 21st September 2022, can be found at:

https://development.standards.ieee.org/myproject-web/public/view.html#pardetail/10027

and the CSD (Criteria for Standards Development) at:

https://mentor.ieee.org/802-ec/dcn/22/ec-22-0083-00-ACSD-p802-1qdt.pdf

extracts of relevant material from the PAR and CSD follow.

PAR Scope, Purpose, and Need

The Scope of the standard (IEEE Std 802.1Q) as amended by this project remains unchanged and is shown below. The Purpose (clause 1.3) of IEEE Sd 802.1Q is not changed by this project.

Scope:

This standard specifies Bridges that interconnect individual LANs, each supporting the IEEE 802 MAC Service using a different or identical media access control method, to provide Bridged Networks and VLANs.

Scope of the Project:

This amendment specifies procedures, protocols and managed objects to enhance Cyclic Queuing and Forwarding, comprising: 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; a protocol for determining the phase relationship between a transmitter's and a receiver's bin boundaries in time; managed objects, Management Information Base (MIB), and YANG modules for controlling these procedures; and an informative annex to provide guidance for applying these procedures. This amendment also addresses errors and omissions in the description of existing IEEE Std 802.1Q functionality.

Purpose:

Bridges, as specified by this standard, allow the compatible interconnection of information technology equipment attached to separate individual LANs.

Need for the Project:

The existing Cyclic Queuing and Forwarding (CQF) functionality in IEEE Std 802.1Q provides bounded end-to-end delays, allows simple delay analysis methods, and does not depend on per-flow state. These properties are critical for scaling up Time-Sensitive Networking to large networks with a high number of simultaneous flows, such as service provider networks. This amendment extends the existing CQF functionality to support long physical links with high delay, processing delay variations in bridges, non-time-synchronized ingress traffic, and/or flows with a wider range of latency requirements. These properties enhance the suitability of CQF for large networks.

CSD broad market potential

The features of this standard broaden the applicability of Time-Sensitive Networking (TSN) to networks with simpler bridges than are possible with the existing, deployed TSN features, and to service provider networks, a large market so far untapped by TSN.

The interest expressed by vendors and users in IEEE 802.1 indicates that sufficient interest will exist outside IEEE 802.1 for this standard to succeed.CSD compatability

CSD technical feasibility

The existing Asynchronous Traffic Shaping and Cyclic Queuing and Forwarding provisions of IEEE Std 802.1Q bound, on either side, the complexity of this standard. Both are deployed, indicating the feasibility of this standard.

Introduction to the current draft

This is an initial draft of P802.1Qdv.

P802.1Qdv/D0.3
July 12, 2023

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

Draft Standard for Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment: Enhancements to Cyclic Queuing and Forwarding

Unapproved draft, prepared by the
Time-Sensitive Networking (TSN) Task Group of IEEE 802.1

Sponsored by the
LAN/MAN Standards Committee
of the
IEEE Computer Society

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

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

---

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

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

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 the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE SA) Standards Board. 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.

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.

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 Standards Coordinating Committees 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 imply 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.

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.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. 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, IEEE does not waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate fee, 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. 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 ten years. When a document is more than ten 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

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.

IEEE SA Website at http://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.

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.

Participants

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

At the time this standard was completed, the IEEE 802.1 working group had the following membership:

Glenn Parsons, *Chair*
Jessy Royer, *Vice Chair*
Janos Farkas, *TSN Task Group Chair*
Lily Lv, *Editor*

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

A.N. Other

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

When the IEEE-SA Standards Board approved this standard on <dd> <month> <year>, it had the following membership:

Chair
Vice-Chair
Past Chair
Secretary

*Member Emeritus

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

Introduction

This introduction is not part of IEEE Std 802.1Qdv-20XX, IEEE Standard for Local and metropolitan area networks—Bridges and Bridged Networks—Amendment: Enhancements to Cyclic Queuing and Forwarding

This standard amends IEEE Std 802.1Q™-2022 as previously amended by IEEE Std 802.1Qcz™-2022. In particular it enhances capabilities introduced by IEEE Std 802.1Qch™-2017.

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

Contents

Figures

Tables

Draft Standard for
Local and Metropolitan Networks —

Bridges and Bridged Networks

Amendment: Enhancements to Cyclic Queuing and Forwarding

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

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

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

The contents of this initial Framemaker 'kit' clause are taken from an SA ballot copy of P802.1Qcz. This provides and example of how to modify 1.1 Scope and 1.3 Introduction. See the EDITOR-PLEASE-READ file and the 802-1Qxx-conditional-tags.fm file in this book for the various conditional tag views of this clause. This paragraph is in paragraph style 'Text', character style 'Editor', with conditional tag 'Editor comment'.

1. Overview

1.3 Introduction

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

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

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

2. Normative references

The contents of this initial Framemaker ‘kit’ clause are taken from an SA ballot copy of P802.1Qcz. This provides an example of how to add to the list of references. See the EDITOR-PLEASE-READ file and the 802-1Qxx-conditional-tags.fm file in this book for the various conditional tag views of this clause. The Final text view may be useful in identifying missing references, failure to update existing references, and avoiding duplicates. New references should be added to the base text in collating order.

Insert the following items into the list of Normative References:

3. Definitions

The contents of this initial Framemaker ‘kit’ clause are taken from an SA ballot copy of P802.1Qcz. This provides an example of how to add to the list of definitions. See the EDITOR-PLEASE-READ file and the 802-1Qxx-conditional-tags.fm file in this book for the various conditional tag views of this clause. The Final text view may be useful in identifying missing definitions, failure to update existing definitions, and avoiding duplicates. New definitions should be added to the base text in collating order. Before adding new definitions, or modifying existing definitions search for all (if any) text conditionally tagged ‘Delete’ or ‘Change remove’ and delete it (use Edit>Find/Change...Conditional Text, and in the ‘Find Conditional Text’ pop up select the tag to be found, and then Change ‘To Text’ leaving the change to field blank and selecting ‘Change All’). Then convert (using the Conditional Tags panel) any Change add or Insert tagged text to ‘Base hide’ (re remove tags entirely from it, if it is to be shown).

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

4. Abbreviations

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

The contents of this initial Framemaker ‘kit’ clause are taken from an SA ballot copy of P802.1Qcz. This provides and example of how to modify this clause. See Clause 3 for preliminary steps before editing for your amendment.

CPAP Cyclic queuing and forwarding (CQF) Phase Alignment Protocol

5. Conformance

5.4 VLAN Bridge component requirements

5.4.1 VLAN Bridge component options

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

5.4.1.12 Cyclic Queuing and Forwarding (CQF) requirements

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

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

5.4.1.13 Time-based CQF requirements

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

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

5.4.1.14 Count-based CQF requirements

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

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

5.13 MAC Bridge component requirements

5.13.1 MAC Bridge component options

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

5.13.1.4 Cyclic Queuing and Forwarding (CQF) requirements

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

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

5.13.1.5 Time-based CQF requirements

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

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

5.13.1.6 Count-based CQF requirements

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

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

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

5.14 End station requirements for count-based CQF

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

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

8. Principles of Bridge operation

8.6 The Forwarding Process

8.6.5 Flow classification and metering

Change 8.6.5 as follows:

<< Editor’s note: There are a great many references to ATS in 8.6.5, because of the split between “General flow classification and metering” 8.6.5.1 and “per-Stream classification and metering” 8.6.5.2, which was introduced by ATS. The editor believes that CQF will want to use 8.6.5.2 per-Stream C&M. Perhaps we can present CQF as a special case of ATS to minimize changes. Perhaps we change lots of references in 8.6.5 from “ATS” to “ATS or CQF”. Perhaps we invent a new term that encompasses both ATS and CQF for use in subclauses 8.6.5.2 through 8.6.5.4.. Comments are solicited.>>

<< Editor’s note: It is not settled whether, in Clause 8, CQF should use bin number or eligibility time when queuing frames. Using bin number simplifies the CQF bin assignment clauses somewhat, but likely means less reuse of the existing ATS descriptions, especially in 8.6.8 Transmission selection. Comments are solicited. >>

Insert the following at the end of 8.6.5:

8.6.5.7 CQF time-based bin assignment

<< Editor’s note: It is not settled whether, in Clause 8, CQF should use bin number or eligibility time for storing frames. The title of this subclause may well be, “CQF time-based eligibility time assignment”. >>

Insert the following at the end of 8.6.5:

8.6.5.8 CQF count-based bin assignment

<< Editor’s note: It is not settled whether, in Clause 8, CQF should use bin number or eligibility time for storing frames. The title of this subclause may well be, “CQF count-based eligibility time assignment”. >>

8.6.8 Transmission selection

Insert the following at the end of 8.6.8:

8.6.8.6 CQF transmission selection algorithm

<< Editor’s note: It is not settled whether, in Clause 8, CQF should use bin number or eligibility time for storing frames. If we use eligibility time, there may be no need for this subclause. >>

Insert the following Clause:

99. CQF Phase Alignment Protocol

99.1 Overview of CPAP

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

Figure 99-1 is an abridgment of Figure T-4. It shows an example of CQF. Bridge A is transmitting from a CQF queue to Bridge B. Whether it is using time-based CQF (T.3) count-based CQF (T.4) or some other method to fill its bins is irrelevant. Bridge B is using time-based CQF to assign frames to its output bins (not shown). 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 T.3.2, Bridge B needs to assign each received Stream frame to an output queue bin, based solely on the time of arrival of the frame at Bridge B's input port (IEEE Std 802.3 clause 90). In order to assign frames to bin based only on time, Bridge B runs its output cycles with exactly the same period T_C as Bridge A (see T.3.1), though not necessarily in phase (synchronized). The problem to be solved by CPAP is described in T.3.2.2. Bridge B needs to establish the frame arrival time at its input port that corresponds to a transmit cycle boundary in Bridge A. That is, Bridge B wants to know what input timestamp it would expect to see on a frame Bridge A transmitted at exactly the start of a cycle in Bridge A.

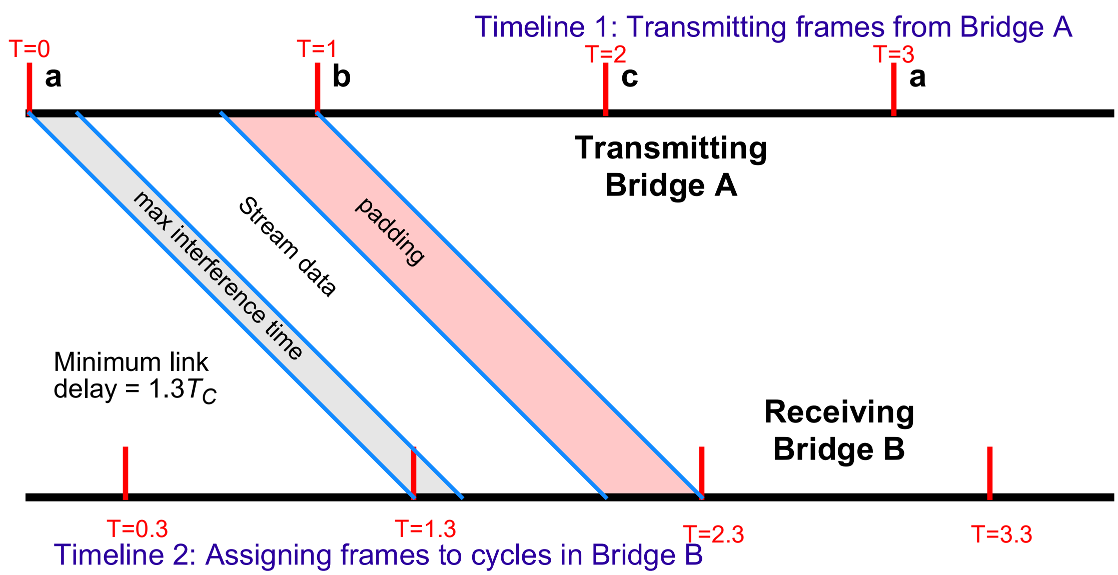

Figure 99-1—Aligning the transmitter and receiver CQF cycle start times

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 input 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 CQF cycles. (Such variations are included in the definition of T_V in T.3.2.)

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 CQF-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.

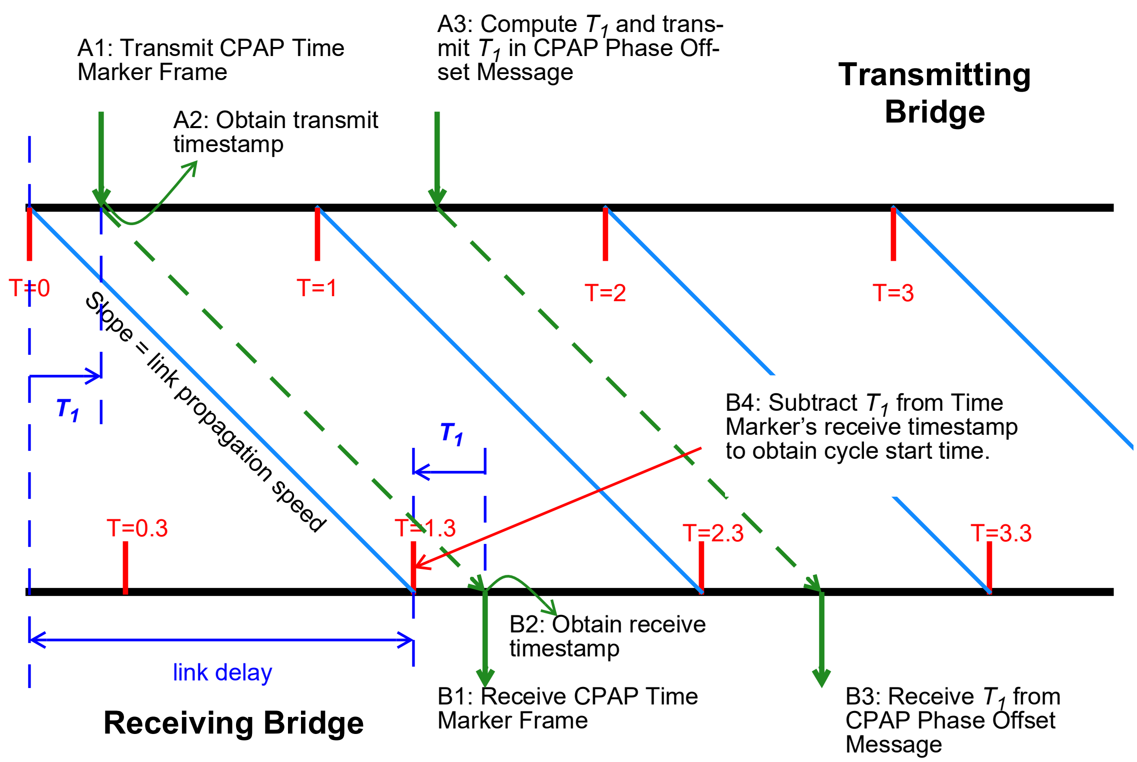

Figure 99-2—CQF Phase Alignment Protocol sequence

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

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 CQF 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 $-T_C \leq T_I \leq T_C$, where T_C is the CQF cycle time. This time can be positive or negative, depending on whether it is compared to a past or a future CQF 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 CQF 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 timing

<< 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. >>

<< Editor's note. In the editor's opinion, this document should NOT take an approach, which is possible, of using the CPAP protocol to control the timing of the receiver's *output* cycles. That would be re-inventing IEEE Std 1588,. >>

99.4 CPAP message frame formats

<< Editor's note: Input is required for the editor to finish this section. At least the following possibilities can be identified:

a) We could define an EtherType that would serve for both frames. This is a problem, as EtherTypes are becoming an endangered species.
b) No data is carried by the CPAP Time Marker Frame except its identity as such. We could define a TLV to be carried, for example, in one of the IEEE 802.1AS and/or IEEE 1588 Precision Time Protocol (PTP) frame types. This has the advantage that one or the other of these protocols are often used with TSN, and that they are often used with the IEEE 802.3 timestamp function, but of course this is a disadvantage if neither is used. If PTP is not running, but we use PTP to carry CPAP messages, we must ensure that we do not accidentally start up PTP or trigger other errors.
c) We could piggyback CPAP on LLDP.

>>

<< Editor's note: It seems that a CPAP Phase Offset Message should somehow be tied to a particular previous CPAP Time Marker Frame. We could assume that it applies to the previously-received one, if any. We could add a serial number to the CPAP Time Marker Frame and include it in the CPAP Phase Offset Message. We could include a time interval in the CPAP Phase Offset Message such that the message applies

only to the one CPAP Time Marker Frame received within that interval before receipt of the CPAP Phase Offset Message.

Readers are solicited for their opinion(s) on these possibilities, or for a suggestion for others. >>

99.5 CPAP managed objects

<< Editor's note: To Be Done when the other issues are settled. >>

Insert the following Clause:

100. Cyclic queuing and forwarding

100.1 CQF managed objects

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

100.1.1 Cycle and priority structure managed objects

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

a) The cycle time of the slowest CQF priority value (as a rational number of nanoseconds).
b) The priority value of the slowest CQF cycle.

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

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

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

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

100.1.2 Cycle phase managed objects

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

a) The start time of an instance of the slowest CQF cycle, in terms of the system clock.

This variable establishes the phase of the input or output cycle. Typically, this variable would be manageable the network administrator for output ports. For time-synchronized systems, it can be administered for input ports, as well, in order to adjust for link delay. Alternatively, the input phase can be determined dynamically (Clause 99), and be read-only for the network administrator.

100.1.3 Cycle variation information

For each output port only, we have:

a) The largest offset from the nominal (system clock) Nominal Output Cycle Start time (NOCS, T.3.11.3) event to the actual cycle start time, in the negative (actual earlier than NOCS) direction.
b) The largest offset from the nominal (system clock) NOCS event to the actual cycle start time, in the positive (actual later than NOCS) direction.

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

100.1.4 Dead time

Given the context of dead time determination described in T.3.11.5, the following items are required by CQF:

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

   There is a component of this dead time computed according to T.3.11.5, as well as one computed in item e) of T.3.11.3. The sum of these must be known to the adjacent transmitting port.

b) Per output port, per priority level, the total dead time that is to be provided at the end of each transmit cycle.

   This can be configured, obtained from the adjacent input system, or be a maximum of these values.

c) The allocable bandwidth for this input port and priority level.

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

d) The allocable bandwidth for this output port and priority level.

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

100.1.5 CQF forwarding delays

100.1.5.1 Minimum CQF delay

Read-only per Stream. See T.3.2 and Figure T-4. This is the CQF-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 CQF delay is the earliest time when the bin, into which the frame is stored, could be enabled for output, assuming that the extra delay (100.1.5.3) is 0.

Thus, this time does not include the time required to empty a bin, the link delay, or any extra imposed delay.

100.1.5.2 Maximum CQF delay

Read-only per Stream. See T.3.2 and Figure T-4. This is the CQF-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 CQF delay is the latest possible time when the bin, into which the frame is stored, could be enabled for output, assuming that the extra delay (100.1.5.3) is 0.

Thus, this time does not include the time required to empty a bin, the link delay, or any extra imposed delay.

100.1.5.3 Extra delay

Configurable for each priority level, input/output port pair, and stream_handle, a read-write object is required to specify the number of extra bins, beyond that computed/configured by other means, that frames are to be stored in, in order to increase their delivery delay in this Bridge.

100.2 CQF LLDP TLVs

<< Editor's note: In general, it would be good for two CQF 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. >>

Annex A

(normative)

PICS proforma—Bridge implementations[1]

<< Editor’s note: To Be Done. >>

1

Annex T

(informative)

Cyclic queuing and forwarding

Replace the contents of Annex T with the following:

T.1 Principles of CQF

T.1.1 Overview

Cyclic queuing and forwarding (CQF) 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 output queue using CQF (a "CQF queue") is notionally divided into bins. The bins are enabled for output 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 CQF segment of a network. At any given instant in time, a particular output 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 T.1.2.

b) Each Stream utilizing a CQF 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 CQF queue do not exceed T_C, even including possible interference from other queues on the port. See T.1.3.

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

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 (T.3.9). These properties have significant consequences in larger networks, because they support the aggregation of Streams (T.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 (T.2) to provide CQF facilities for different levels of latency and bandwidth requirements.T.2

T.1.2 CQF transmission selection

A CQF 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 output bin number *b* when queued for output 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 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 output bins (queues) at the same time. This is a perfectly valid method for implementing CQF. The present standard describes CQF as a one or more individual class of service queues, each with multiple bins. This formulation offers a wider range of services.

CQF 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.

T.1.3 Bin selection

When a Stream frame is received and forwarded to a class of service output queue that is enabled for CQF, the frame is assigned to a particular bin in that queue. There are two methods for assigning a frame to a bin, time-based (T.3.1) and count-based (T.4). The same frame can be assigned to bins on two different output 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 output port. It can be configured on an input-output port pair basis. The selection method can be configured for specific Streams.

T.2 CQF in multiple queues on one output port

T.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 output port, as shown in Figure T-1.

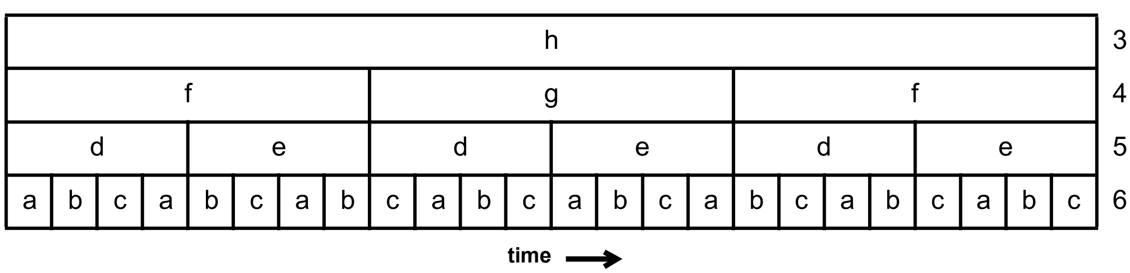

Figure T-1—Multiple T_C values on multiple queues on one CQF port

In Figure T-1, we have a schematic timeline. Four class of service queues have been configured for CQF, 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 T-1 label which bin is output during the cycle. There are 9 bins a through i. Bin i, the second bin at priority 3, is not shown. In this example, priority 6 uses three bins, because the timing is tight; the others use two each.

We assume here that the receiver of a frame can identify the particular CQF 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 T-2, where we illustrate the timing of transmission of frames from three levels of CQF and the best-effort (BE) level. Note that CQF traffic can be delayed within its window by interference from both higher priorities (e.g. the first priority 4 frame) and lower priorities (e.g. the first priority 6 frame), but that it will always get out before the window closes, assuming that the bandwidth is not oversubscribed.

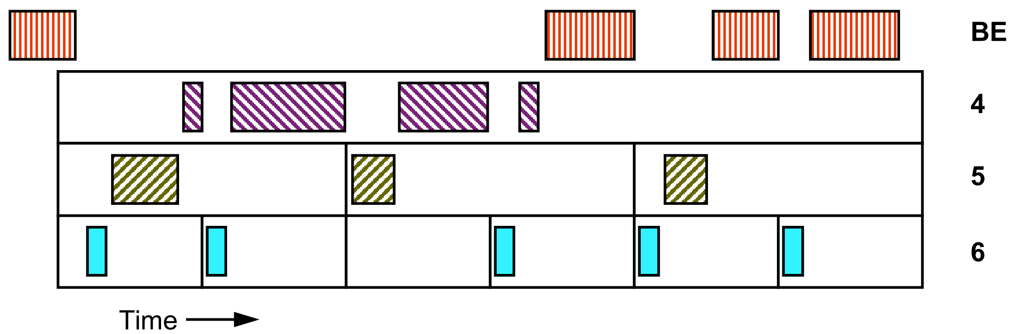

Figure T-2—Transmission timing

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

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

Any such overprovision can equally be thought of as an increased latency for that same Stream. That is, if that oversubscribed Stream was the only Stream, then the T_C cycle time could be shortened to exactly the point of one frame per cycle, with no overprovisioning, and thus give a faster latency. Overprovision = 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.

T.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 T-3.

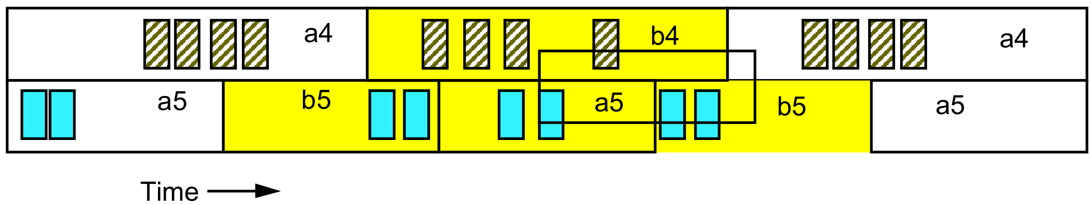

Figure T-3—Variable T_C

In Figure T-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 T.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 output 6 frames over the course of cycle b4, thus taking up 60% of the bandwidth during that period. There is, therefore, 110% of the bandwidth that must be output during the period that b4 is transmitting. b4 cannot output all of its data. Some of it must be somehow delayed, but there is no place to put that data. Deterministic QoS is not obtained.

Having an integral number of cycles at each layer fitting exactly into one cycle at the next-slower layer ensures that the lower-priority, slower cycle, will always have sufficient time to output all of its frame, because the problem in Figure T-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 T-1. This scheme also bounds the number of preemption events that can steal bandwidth from a given priority level (see T.3.3).

T.2.3 Admission control for multiple T_C values

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

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

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

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

T.3 Time-based CQF

T.3.1 Frequency lock requirement

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

T.3.2 Timeline for time-based bin assignment

We have two Bridges, A and B. Both are running time-based CQF on each of multiple ports.

When a CQF 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 T-4 shows an example of CQF. 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 T-4, we use the following notation for time intervals:

T_C nominal (intended) period of the CQF 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 output from B

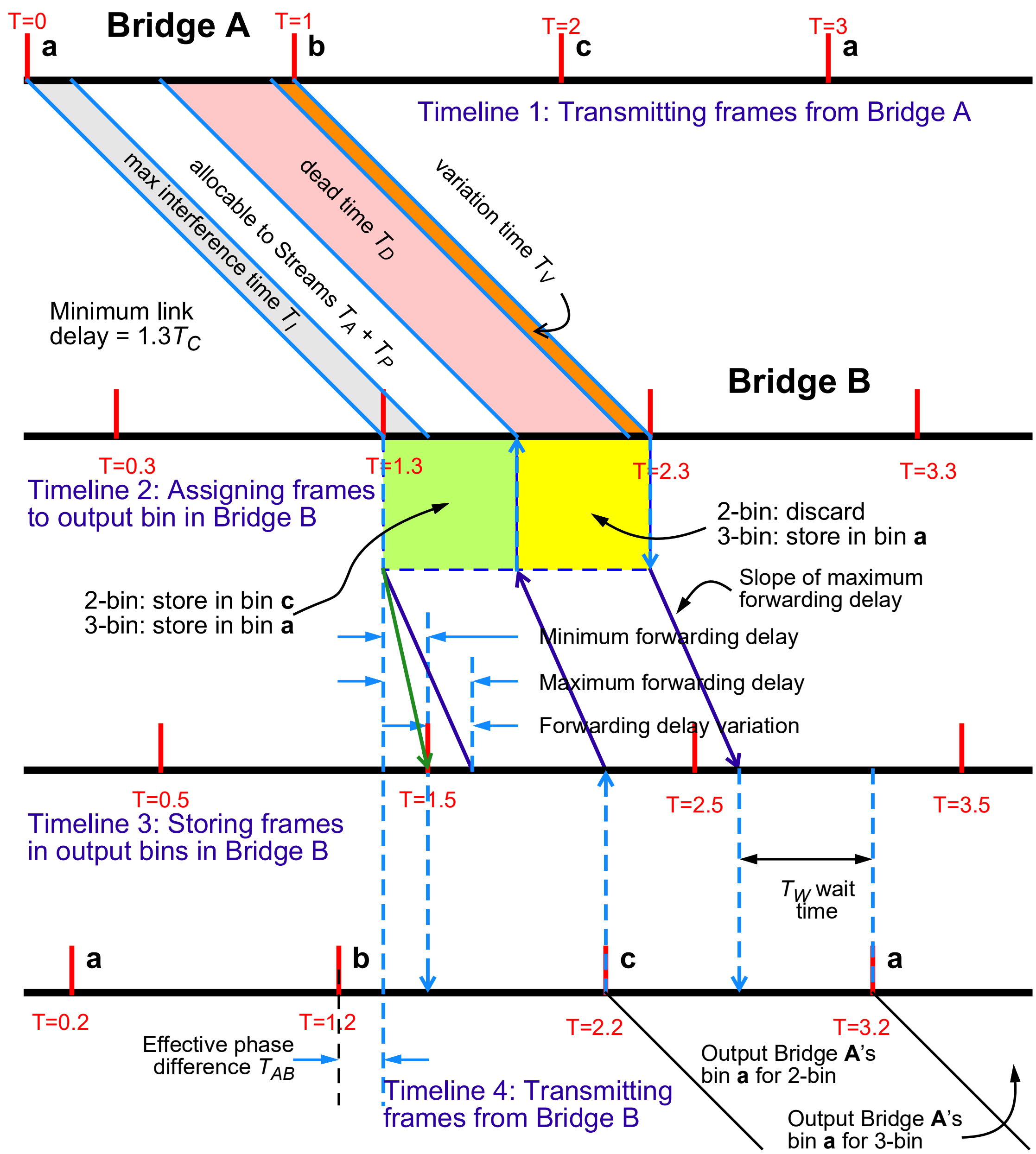

Figure T-4—Example of timelines for time-based CQF

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 T-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 output bins, the ticks are the points in time when the transmission gate of one queue is closed, and the next queue's transmission gate is opened. These are the points in time as programmed into the managed objects that control ETS. An implementation may need to schedule cycle start times in anticipation of the time specified in the managed objects in order to maximize throughput. Note that the preamble of an IEEE Std 802.3 Ethernet frame can be transmitted before the start of a cycle.

T.3.2.1 Output timeline 1

Figure T-4 shows an interference delay T_I (the gray area) between the start of Bridge A's cycle (the red ticks in Figure T-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 T-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 an CQF Stream. This value is always bounded. See T.3.3.

There can be some variation in the time from the selection of a frame for output 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 CQF 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 T.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 output 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 output 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.

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

T.3.2.2 Receive timeline 2

The timeline at the receiving port is timeline 2 in Figure T-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 output 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 output timeline. This offset is shown as T_{AB} in Figure T-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 T-4 can be computed. The phasing of the Bridges' output bin cycles affects the end-to-end latency of any Stream, so that phasing must be known when the end-to-end latency is computed. However, the end-to-end latency is not necessarily an integer multiple of the cycle time, 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 T-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.

T.3.2.3 Storing frames timeline 3

The timeline at the point where frames are stored into an output bin is timeline 3 in Figure T-4. The red ticks on timeline 3 mark the earliest point at which the first frame transmitted from a particular bin could reach the output bins (neglecting transmission time on the input medium). These ticks are offset from timeline 2 by the minimum forwarding delay, required to forward the frame from the input port to the output queue. The maximum forwarding delay is also shown. The forwarding delays shown in Figure T-4 include the time to install the frame in the output bin and for its presence to filter through to the point that it can be selected for output.

For a Bridge B that is connected to and receiving CQF 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 T-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 T-4 show the time windows for bin assignment. If two output 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.

Time T_W in Figure T-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 T-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 T-4 could, in theory, be longer than one cycle time T_C. See T.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 (T.2.4) can be met.

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

T.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 T.3.1 and Figure T-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 T-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 T-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.

T.3.4 T_C computation

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

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

Annex T CQF assumes the 2-bin scheme, and so assumes that T_D and T_V are small enough and T_C large enough to leave a useful T_A. Assuming that one's goal is the smallest possible T_C:

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

T.3.5 Calculation of dead time T_D

Timeline 3 in Figure T-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 output cycle starts (the tick on timeline 4), moved backward by the worst-case forwarding delay. This is the last moment on timeline 3 that a frame can be assigned to bin c in the example in Figure T-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/output port pair on a particular Bridge computes a value for T_D that exceeds this maximum, then 3-bin operation is required.

T.3.6 More than 3 output bins

So far, the discussion of Figure T-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 output bins, and assign received frames to bins whose output is scheduled far enough ahead in time to ensure that, in the worst case, they will arrive in the 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 output 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 output bins can be allocated, and used to impose a delay of an arbitrary number of cycle times T_C on every frame.

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

T.3.7 Deterministic behavior of time-based CQF

CQF guarantees the Deterministic QoS by the following argument.

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

We consider only one value of T_C along the path of a given Stream from Talker to Listener. T.2.3 and T.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 T.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 CQF 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 CQF output bin after entry to the network. That is, the frames from a given Stream do not exceed the Talker contract in the first CQF output bin in the network.

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

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

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

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

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

In all other cases, when a Stream changes cycle times, the Stream must pass through a conditioning step, such as a count-based CQF step (see T.4.4), to ensure that the Stream never exceeds its contract in the new cycle time.

T.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 T.3.1. Referring to Figure T-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 output window in which a frame received in that input window will be transmitted. This is the dwell time for the frame in this Bridge. A maximum and minimum time for this delay is given in 100.1.5.

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 CQF 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 T-4, we can see that the difference between using two and three bins for a given input-output port pair is really a matter of rounding up the link delay to an integral number of cycle times. If the sum of link delay and phase delay between output cycles is negligible, or happens to be very nearly an integer multiple of the cycle time, then the yellow "discard" area is small, and two bins can be used. If sum is larger, then one necessarily chooses between a smaller allocation (large discard area) and increased delay.

T.3.10 Output bin selection

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

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

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

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

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

Where P_{io} is the cycle phase offset from input port i to output port o and B_n is the number of output bins on the port. See T.3.11.4 for the determination of P_{io}. Note that in the extreme case of all output ports using two bins, all synchronized, and all input cycles in phase with the output cycles, the table P_{io} reduces to a single value, 0 or 1.

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 CQF service guarantees.

T.3.11 Parameterization of time-based CQF

Let us go through the exercise of initializing an input/output port pair for CQF. 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 CQF.

T.3.11.1 Cycle wander

Adjacent Bridges must be frequency locked as described in T.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 T.3.2.1.

T.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.

T.3.11.3 Calculating the number of bins required

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

a) Establish a Nominal Input Cycle Start time (NICS) for the input port, and a Nominal Output Cycle Start time (NOCS) for the output port. The NICS and NOCS each repeat every T_C seconds, according to the system clock. We will assume that the offset between them is a constant (i.e., they are both driven by the same system clock).
b) Compute the earliest time, relative to the NICS, at which the first frame of a cycle can receive its IEEE Std 802.3 clause 90 timestamp. This frame is assumed to be a minimum-length frame (64 bytes plus overhead).
c) Compute the earliest time, relative to the NICS, at which a bin on the output port must be eligible to receive the frame. This is equal to the timestamp time in bullet b) plus the minimum time required to move the frame through the Bridge to the output bin.
d) Compute the latest time, relative to the NICS, at which the last frame of a cycle can receive its timestamp. This frame is assumed to be a minimum-length frame.
e) If the difference between the earliest timestamp and the latest timestamp is greater than or equal to the cycle time T_C, then dead time must be imposed on the transmitter, at the end of the cycle, to reduce the difference.
f) Compute the latest time, relative to the NICS, at which the last frame of a cycle can be stored into an output bin and be ready for selection for transmission, given the worst-case forwarding delay through the Bridge.
g) Convert these earliest b) and latest d) arrival times to times relative to the NOCS of the output port.
h) Arbitrarily label an input port NICS event NICS0. Determine the latest subsequent NOCS event, which we will label NOCS0, during which the earliest-arriving frame of NICS0 must be stored in the output queue.
i) Determine the earliest subsequent NOCS event, which we will label NOCSn, before which the latest-arriving frame from NICS0 can be stored in the queue, and still be available for transmission at the start of cycle NOCSn.
j) The number of cycles NOCS0 through NOCSn, inclusive, is the number of bins required for the input/output port pair, B_{io}.

The number of bins required can sometimes be reduced by:

— Imposing a larger dead time on the transmitter feeding the input port, at the end of every cycle;
— Altering the phase of the output port's cycle; and/or
— Imposing implementation-specific limitations on the Streams, e.g. reducing fan-in to an output port, or restricting bridging/routing features to reduce forwarding delay variation.

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

T.3.11.4 Initial bin phase

The number of bins required on an output port is the maximum required over all input ports. This may be further increased by intentional delays (T.3.6). When initializing an input port, a correspondence must be made between the input and output ports, so that a frame received on the input port will be stored in a particular bin in the output port, the one that will become the transmitting bin in the appropriate number of output cycles in the future.

The phasing between input and output ports' cycles, and thus the number of bins in port *o* used by port *i*, is determined by the P_{io} table defined in T.3.10. We compute P_{io} when initializing CQF, or when the relative phase of the input and output ports change significantly, by selecting a time T that coincides with the start of an input cycle on input port *i* and computing:

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

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

T.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 output port. Increasing the dead time can reduce the number of bins required, and thus the end-to-end latency of a data Stream, as described in T.3.11.3. There are, at the very least, the following ways to make this decision:

— Configure the output cycle phase and number of bins to use for all Bridges, in order to establish a constant per-hop delay in a network with short links. Let each system compute the dead time on each input port required to make this work, and the bandwidth available for allocation. Convey the required dead time either by protocol or by management to the transmitters, and the available bandwidth to the admission control system.

— Configure the output cycle phase on all Bridges. Configure minimum and maximum allocable bandwidth values for each CQF priority level. Let each system compute the minimum number of bins required to meet the minimum bandwidth value, taking advantage of the maximum bandwidth value to compute a dead time value that minimizes the number of bins required. This would be useful in a network with very long links. Convey the resultant dead time to the transmitter via protocol, and the resultant allocable bandwidth to the admission control system.

— Using data sheet information, configure all parameters via network management. Adjust the output port cycle phasing to optimize the delay for certain specific Streams.

T.4 Count-based CQF

As described in T.3.1, time-based bin assignment assigns frames to output bins based on the time of arrival of the frame, and requires that the output queues of successive hops along an CQF path run at exactly the same frequency, in order to ensure that no bin's capacity can be exceeded. This requirement can be relaxed, at the cost of implementing a state machine for each Stream passing through each output port. Then, the output queues along the path can run nearly the same frequency, and their relative phases (T_{AB} in Figure T-4) can diverge.

Count-based bin assignment (count-based CQF) is an alternative description of the paternoster algorithm defined in [B1]. 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 CQF 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.

T.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 T.3.2.1 and Figure T-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 (T.3.2.1, T.3.3), T_P is the penalty incurred if this priority level is preemptable (T.3.3), is the T_D is the dead time imposed by Bridge to which this Bridge is transmitting (T.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 T.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 output 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.

T.4.2 Dead time T_D

Count-based CQF in a receiving Bridge cannot impose dead time (T.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.

T.4.3 Number of output bins

In an ideal world, only two bins are required per output 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 T-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 CQF. 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.

T.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 CQF, and that has reserved bin space allocated for it, will not disrupt the deterministic behavior of CQF. However, an CQF Bridge could receive input from a Bridge, a Talker, a router, or any other device that uses some deterministic algorithm(s) to condition its Streams, but uses an algorithm other than CQF. We assume that reservations (contracts) for these Streams can be translated into CQF 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 CQF, we cannot use just a frame's arrival time to assign it to a bin, except by overprovisioning CQF sufficiently to accommodate the worst-case burst behavior of the algorithm employed by the sender. We would like to accommodate such input.

The count-based bin assignment makes this possible. A Bridge uses the same bin structure and output methods described for time-based bin assignment in T.3.1, but instead of obtaining the bin selector S from the time of receipt of the input frame, as described in T.3.10, it uses a state machine dedicated to each CQF 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 output bin can accept input from both time-based and count-based Streams, as long as they share the same cycle time; separate count-based and time-based bins or queues are not necessary. In addition, a paranoid network administrator could very well configure count-based bin assignment on every Stream in a frequency-locked network, in order to guard against misbehaving Bridges or Talkers. That is, while count-based bin assignment can be thought of as separate algorithm from time-based bin assignment, it can also be thought of as a protection mechanism for time-based assignment that can be employed as need, and when employed everywhere, removes the restriction that Bridges operate at exactly the same frequency.

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

T.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 (T.6.1). Ideally, his higher-bandwidth Stream can be assigned a CQF 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 T-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 CQF, placing one frame from each stream into each output bin. In Bridge B, the three Streams are aggregated into a single aggregated Stream. This Stream is forwarded through Bridges C and D. Bridge E dissolves the aggregated Stream, distributing the component Streams' frames on three ports.

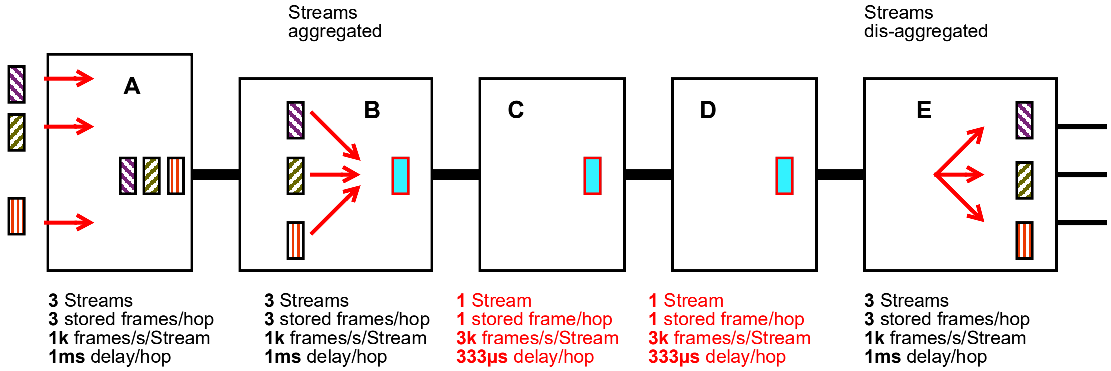

Figure T-5—CQF Stream Aggregation example

As indicated in the captions in Figure T-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 T.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 T.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 T.5.4.

T.5.1 CQF Stream aggregation

Stream aggregation usually includes changing the cycle time from a slower to a faster cycle. Changing the cycle time is discussed in T.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 outputting 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 output port). In the worst case, the same number of output bins are required on the output port as would have been needed had there been no aggregation.

T.5.2 CQF Stream disaggregation

Stream disaggregation usually includes changing the cycle time from a faster to a slower cycle. Changing the cycle time is discussed in T.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 us time to select the output 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 output bins in the normal manner.

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 T-5) can introduce a small amount of delay variation. Any such variation increases the buffer requirements in the disaggregating Bridge. See T.5.3.

T.5.3 CQF 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 T-5 use Asynchronous Traffic Shaping, instead of CQF, 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 (T.5.2) or mixing Bridge (T.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 T-5), as its output cycles are not necessarily in phase with the aggregating Bridge's input cycles.

T.5.4 CQF 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. Rather,

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 output bin appropriate for its output port, whether nor not it is also being reaggregated. As for the case for disaggregation (T.5.2) the output buffer space required depends on the cycle times of the original, component Streams at the time they entered the aggregation.

T.6 Additional considerations

T.6.1 Computing the CQF reservation

At the lowest level, e.g. in a count-base bin assignment state machine, CQF 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 CQF, and also how to combine multiple Streams' TSN and/or CBS specs into a single CQF spec. >>

T.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 non-TSN data, so overprovisioning may not be a serious concern.

Another approach to increased resource utilization efficiency is to run each CQF 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[2] into large transmission units.)

T.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 (T.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.)

T.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.

[2]Not to be confused with Link Aggregation

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.

T.6.5 CQF and credit-based shaper

Looking at Figure T-2, we see that, once the major cycle at priority level 4 begins transmitting, the best-effort traffic is interrupted until all of the CQF level 4 data is transmitted. At some point, as the amount of traffic in a very slow CQF 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 CQF cycle(s). However, the parameters of this shaper must be adjusted as the load on the slow CQF 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.

T.6.6 Interactions among CQF, 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 CQF queue, one must be sure that, in the worst case, the CQF queue can still empty its bin within the alloted period (T_A in T.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.

‘

Annex ZY

(informative)

Bibliography

This chapter should include the .1Q bibliography in total, with the proper conditional text tags.

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

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

[B2] Finn, Norman, https://mentor.ieee.org/802.1/dcn/21/1-21-0056-00-ICne-input-synchronization-for-cyclic-queueing-and-forwarding.pdf

Annex ZZ

(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. >>