



**IEEE Standard for
Local and Metropolitan Area Networks—
Bridges and Bridged Networks
Amendment 34:
Asynchronous Traffic Shaping**

IEEE Computer Society

Developed by the
LAN/MAN Standards Committee

IEEE Std 802.1Qcr™-2020

(Amendment to IEEE Std 802.1Q™-2018 as amended by
IEEE Std 802.1Qcp™-2018, IEEE Std 802.1Qcc™-2018,
IEEE Std 802.1Qcy™-2019, and IEEE Std 802.1Qcx™-2020)

IEEE Std 802.1Qcr™-2020

(Amendment to IEEE Std 802.1Q™-2018 as amended by
IEEE Std 802.1Qcp™-2018, IEEE Std 802.1Qcc™-2018,
IEEE Std 802.1Qcy™-2019, and IEEE Std 802.1Qcx™-2020)

IEEE Standard for Local and Metropolitan Area Networks—

Bridges and Bridged Networks

Amendment 34: Asynchronous Traffic Shaping

Developed by the

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 24 September 2020

IEEE SA Standards Board

Abstract: Procedures and managed objects for Bridges and end stations to perform Asynchronous Traffic Shaping over full-duplex links with constant bit data rates are specified in this amendment to IEEE Std 802.1Q™-2018.

Keywords: amendment, Bridged Local Area Network, IEEE 802.1Q™, IEEE 802.1Qcr™, LAN, Local Area Network, MAC Bridge, Metropolitan Area Network, traffic shaping, Virtual Bridged Local Area Network, Virtual LAN

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

Copyright © 2020 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 6 November 2020. Printed in the United States of America.

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

PDF: ISBN 978-1-5044-7100-8 STD24430
Print: ISBN 978-1-5044-7101-5 STDPD24430

IEEE prohibits discrimination, harassment and bullying.

For more information, visit <https://www.ieee.org/about/corporate/governance/p9-26.html>.

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

Important Notices and Disclaimers Concerning IEEE Standards Documents

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

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

IEEE Standards documents are developed within 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](#). An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.

Laws and regulations

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

Data privacy

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

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under US and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, IEEE does not waive any rights in copyright to the documents.

Photocopies

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

Updating of IEEE Standards documents

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

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

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#). 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](#). 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](#).

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

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

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

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

Glenn Parsons, *Chair*
John Messenger, *Vice Chair*
Jessy V. Rouyer, *Secretary*
János Farkas, *Chair, Time-Sensitive Networking Task Group*
Craig Gunther, *Vice Chair, Time-Sensitive Networking Task Group*
Johannes Specht, *Editor*

Astrit Ademaj
Ralf Assmann
Christian Boiger
Paul Bottorff
Radhakrishna Canchi
Feng Chen
Weiyang Cheng
Paul Congdon
Rodney Cummings
Josef Dorr
Hesham Elbakoury
Thomas Enzinger
Donald Fedyk
Norman Finn
Geoffrey Garner
Marina Gutierrez
Stephen Haddock
Mark Hantel
Jerome Henry
Marc Holness
Satoko Itaya
Yoshihiro Ito

Michael Karl
Stephan Kehrer
Randy Kelsey
Hajime Koto
James Lawlis
Christophe Mangin
Scott Mansfield
Kenichi Maruhashi
David McCall
Larry McMillan
Tero Mustala
Roy Myers
Hiroki Nakano
Bob Noseworthy
Tomoki Ohsawa
Hiroshi Ohue
Donald R. Pannell
Michael Potts
Dieter Proell
Wei Qiu
Karen Randall

Maximilian Riegel
Atsushi Sato
Frank Schewe
Mick Seaman
Maik Seewald
Ramesh Sivakolundu
Marius Stanica
Guenter Steindl
Karim Traore
Balazs Varga
Hao Wang
Tongtong Wang
Xinyuan Wang
Karl Weber
Brian Weis
Ludwig Winkel
Jordon Woods
Takahiro Yamaura
Xiang Yu
Nader Zein
William Zhao
Helge Zinner

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

Thomas Alexander
Butch Anton
Stefan Aust
Christian Boiger
Vern Brethour
Matthew Brown
Demetrio Bucaneg
William Byrd
Paul Cardinal
Steven Carlson
Juan Carreon
Pin Chang
Paul Congdon
Charles Cook
Todor Cooklev
Rodney Cummings
Janos Farkas
Avraham Freedman
Craig Gunther
Stephen Haddock
Mark Hantel
Marco Hernandez
Werner Hoelzl
Oliver Holland
Marc Holness

Russell Housley
Woojung Huh
Atsushi Ito
Raj Jain
Sangkwon Jeong
Pranav Jha
Lokesh Kabra
Stephan Kehrer
Randy Kelsey
Stuart Kerry
Evgeny Khorov
Yongbum Kim
Hyeong Ho Lee
Jennifer Lemon
James Lepp
Jon Lewis
Arthur Marris
Michael Maytum
Brett McClellan
Jonathon McLendon
Larry McMillan
Nick S. A. Nikjoo
Satoshi Obara
Bansi Patel

Walter Pienciak
Clinton Powell
Alon Regev
Maximilian Riegel
Robert Robinson
Jessy V. Rouyer
Chester Sandberg
Frank Schewe
Reinhard Schrage
Michael Seaman
Johannes Specht
Thomas Starai
Walter Struppler
Mark-Rene Uchida
James Van De Ligt
John Vergis
George Vlantis
Lisa Ward
Hung-Yu Wei
Scott Willy
Andreas Wolf
Chi Xu
Yu Yuan
Oren Yuen
Nader Zein

When the IEEE SA Standards Board approved this amendment on 24 September 2020, it had the following membership:

Gary Hoffman, *Chair*
Jon Walter Rosdahl, *Vice Chair*
John D. Kulick, *Past Chair*
Konstantinos Karachalios, *Secretary*

Ted Burse
Doug Edwards
J. Travis Griffith
Grace Gu
Guido R. Hiertz
Joseph L. Koepfinger*

David J. Law
Howard Li
Dong Liu
Kevin Lu
Paul Nikolic
Damir Novosel
Dorothy Stanley

Mehmet Ulema
Lei Wang
Sha Wei
Philip B. Winston
Daidi Zhong
Jingyi Zhou

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 802.1Qcr-2020, IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 34: Asynchronous Traffic Shaping.

This amendment to IEEE Std 802.1Q-2018 specifies procedures and managed objects for Bridges and end stations to perform Asynchronous Traffic Shaping over full-duplex links with constant bit data rates. Asynchronous Traffic Shaping can be modeled as a layer of shaped first-in-first-out (FIFO) queues that are merged into per traffic class FIFO queues in transmission ports. The required minimum number of shaped FIFO queues is adjustable and at least the number of reception ports of a particular bridge.

The amendment specifies an information model for the capabilities of Asynchronous Traffic Shaping. It further specifies a YANG data model based on that information model to support configuration and status reporting. It further defines the relationship between the models introduced by this amendment and the models in the base standard (i.e., IEEE Std 802.1Q-2018 as amended specifically by IEEE Std 802.1Qcp-2018 and IEEE Std 802.1Qcx-2020).

Additionally, this amendment provides an informative framework for worst-case delay analysis in static networks with static configurations.

This amendment also addresses errors and omissions to existing features and includes syntax highlighting in the YANG module listings for easier human readability.

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
USA

Contents

1. Overview.....	13
1.3 Introduction.....	13
4. Abbreviations.....	14
5. Conformance.....	15
5.4 VLAN Bridge component requirements.....	15
5.13 MAC Bridge component requirements.....	15
5.27 End station requirements—PSFP.....	16
5.32 End station requirements—ATS.....	16
8. Principles of Bridge operation.....	17
8.6 The Forwarding Process.....	17
12. Bridge management.....	35
12.4 Bridge Management Entity.....	35
12.31 Managed objects for per-stream classification and metering.....	36
17. Management Information Base (MIB).....	48
17.2 Structure of the MIB.....	48
17.3 Relationship to other MIBs.....	52
17.7 MIB modules.....	52
37. Enhanced Transmission Selection (ETS).....	76
37.3 ETS algorithm.....	76
48. YANG Data Model.....	77
48.3 IEEE 802.1Q YANG model.....	77
48.4 Structure of the YANG model.....	83
48.5 YANG data scheme tree definitions.....	85
48.6 YANG modules.....	90
49. Asynchronous Traffic Shaping (ATS) in end stations.....	133
49.1 Talker transmission behavior.....	133
49.2 Scheduler parameter consistency.....	134
Annex A (normative) PICS proforma—Bridge implementations.....	135
A.5 Major capabilities.....	135
A.14 Bridge Management.....	135
A.46 Per-stream filtering and policing.....	136
A.48 YANG.....	136
A.49 Asynchronous Traffic Shaping.....	137
Annex B (normative) PICS proforma—End station implementations.....	138
B.5 Major capabilities.....	138
B.17 Per-stream filtering and policing.....	138
B.18 Asynchronous Traffic Shaping.....	139

Annex T (informative) Cyclic queuing and forwarding	140
T.2 An approach to CQF implementation	140
T.3 Use of per-stream filtering and policing for CQF	140
T.5 Timing considerations	142
Annex V (informative) Asynchronous Traffic Shaping delay analysis framework	143
V.1 General assumptions	143
V.2 End-to-end delay modeling approach	143
V.3 Buffering delays	144
V.4 Media-dependent delays	146
V.5 Bridge—Internal arrival time recognition delays	146
V.6 Bridge—Internal processing delays	146
V.7 Bridge—Internal clock offset variations	146
V.8 Inter-device clock rate deviations	147
V.9 Combined delay bounds	147
Annex W (informative) Bibliography	149

Figures

Figure 8-13	Flow classification and metering	18
Figure 8-14	Per-stream classification for PSFP	19
Figure 8-15	Per-stream classification and metering for ATS	21
Figure 8-17	Frame timing at gate-close events.....	27
Figure 12-7	Timing points for Stream Gates	42
Figure 48-5	Bridge port model	78
Figure 48-7	TPMR port model	79
Figure 48-9	Provider Edge Bridge C-VLAN Interface model	80
Figure 48-10	Provider Edge Bridge S-VLAN interface model	81
Figure 48-17	Stream filters and stream gates model	82
Figure 48-19	Asynchronous Traffic Shaping model	83
Figure T-1	Example Stream Filter and Stream Gate configuration for CQF	141
Figure T-2	Example Stream Filter and Stream Gate configuration with two values of T	142
Figure V-1	Path of Frames along a single hop with index k with two Bridges	144

Tables

Table 8-4	Stream gate control operations	23
Table 8-6	Transmission selection algorithm identifiers.....	27
Table 8-8	Scheduled Traffic and Stream Gate procedures/variables	29
Table 12-2	Port table entry	35
Table 12-34	The Stream Parameter Table.....	36
Table 12-35	Stream Filter Instance Table	37
Table 12-36	The Stream Gate Instance Table.....	40
Table 12-37	The Flow Meter Instance Table	43
Table 12-38	The Scheduler Instance Table	44
Table 12-39	The Scheduler Group Instance Table.....	45
Table 12-40	The Scheduler Port Parameter Table	45
Table 12-41	The Timing Characteristics Table.....	46
Table 17-30	IEEE8021-PSFP-MIB Structure and relationship to this standard.....	48
Table 48-1	Summary of YANG modules	83
Table 48-9	YANG module dependencies for the stream filters and stream gates model	84
Table 48-11	YANG module dependencies for the ATS model	84

IEEE Standard for Local and Metropolitan Area Networks—

Bridges and Bridged Networks

Amendment 34: Asynchronous Traffic Shaping

(This amendment is based on IEEE Std 802.1Q™-2018 as modified by IEEE Std 802.1Qcp™-2018, IEEE Std 802.1Qcc™-2018, IEEE Std 802.1Qcy™-2019, and IEEE Std 802.1Qcx™-2020.)

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

Editing instructions are shown in ***bold italic***. 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 either by using ~~striketrough~~ (to remove old material) and underscore (to add new material). ***Delete*** removes existing material. ***Insert*** adds new material without disturbing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. ***Replace*** is used to make changes in figures or equations by removing the existing figure or equation and replacing it with a new one. Editing instructions, change markings, and this NOTE will not be carried over into future editions because the changes will be incorporated into the base standard.¹

1. Overview

1.3 Introduction

Insert the following list item after item ck) in the lettered list of 1.3, and reletter the remaining list items accordingly:

- cl) Allow for Asynchronous Traffic Shaping (ATS) over full-duplex links with constant bit data rates.

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

4. Abbreviations

Insert the following abbreviation into Clause 4 in alphabetic order:

ATS Asynchronous Traffic Shaping

5. Conformance

5.4 VLAN Bridge component requirements

5.4.1 VLAN Bridge component options

5.4.1.8 Per-stream filtering and policing (PSFP) requirements

Change 5.4.1.8 as shown:

A VLAN Bridge component implementation that conforms to the provisions of this standard for PSFP shall

- a) Support PSFP as specified in ~~8.6.5.1~~[8.6.5.2.1](#) and ~~8.6.6.1~~[8.6.6 items d\) and e\)](#).
- b) Support the state machines for stream gate control as specified in 8.6.10.
- c) Support the management entities for PSFP as specified in 12.31.

Insert the following subclause (5.4.1.10) after 5.4.1.9:

5.4.1.10 Asynchronous Traffic Shaping (ATS) requirements

A VLAN Bridge component implementation that conforms to the provisions of this standard for ATS shall

- a) Support per-stream classification and metering for ATS as specified in 8.6.5.2.2.
- b) Support queuing with support for stream gates as specified in 8.6.6 items d) and e).
- c) Support the ATS transmission selection algorithm as specified in 8.6.8.5.
- d) Support the ATS scheduler state machines as specified in 8.6.11.
- e) Support the management entities for ATS as specified in 12.31.

5.13 MAC Bridge component requirements

5.13.1 MAC Bridge component options

5.13.1.1 Per-stream filtering and policing (PSFP) requirements

Change 5.13.1.1 as shown:

A MAC Bridge component implementation that conforms to the provisions of this standard for PSFP shall

- a) Support PSFP as specified in ~~8.6.5.1~~[8.6.5.2.1](#) and ~~8.6.6.1~~[8.6.6 items d\) and e\)](#).
- b) Support the state machines for stream gate control as specified in 8.6.10.
- c) Support the management entities for PSFP as specified in 12.31.

Insert the following subclause (5.13.1.3) after 5.13.1.2:

5.13.1.3 Asynchronous Traffic Shaping (ATS) requirements

A MAC Bridge component implementation that conforms to the provisions of this standard for ATS shall

- a) Support per-stream classification and metering for ATS as specified in 8.6.5.2.2.
- b) Support queuing with support for stream gates as specified in 8.6.6 items d) and e).
- c) Support the ATS transmission selection algorithm as specified in 8.6.8.5.

- d) Support the ATS scheduler state machines as specified in 8.6.11.
- e) Support the management entities for ATS as specified in 12.31.

Change the title and text of 5.27 as shown:

5.27 End-station requirements—PSFP

An end-station implementation that conforms to the provisions of this standard for PSFP shall

- a) Support PSFP as specified in ~~8.6.5.1 and 8.6.6.1~~ 8.6.5.2.1.
- b) Support the state machines for stream gate control as specified in 8.6.10.
- c) Support the management entities for PSFP as specified in 12.31.

Insert the following subclause (5.32) after 5.31:

5.32 End station requirements—ATS

An end station implementation that conforms to the provisions of this standard for ATS shall

- a) Support a minimum of two traffic classes, of which
 - 1) One traffic class supports the strict priority transmission selection algorithm (8.6.8.1) and
 - 2) One traffic class supports the ATS transmission selection algorithm (8.6.8.5).
- b) Support the ATS talker transmission behavior, as specified in 49.1.

8. Principles of Bridge operation

8.6 The Forwarding Process

8.6.5 Flow classification and metering

Delete the text of 8.6.5 (as published in IEEE Std 802.1Q-2018). This deletion includes the introductory text of 8.6.5, 8.6.5.1 through 8.6.5.3, Figure 8-13, and Table 8-4.

Insert the following material in place of the deleted material described above. This insertion includes new introductory text for 8.6.5, new 8.6.5.1 through 8.6.5.6, new Figure 8-13 through Figure 8-15, and a revised Table 8-4. Renumber the subsequent figures in Clause 8 accordingly.

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

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

PSFP and ATS share common per-stream classification and metering elements, as shown in Figure 8-13. The stream identification function specified in IEEE Std 802.1CB-2017 can be used to associate received frames with these elements.

8.6.5.1 General flow classification and metering

Bridges that implement general flow classification and metering can identify subsets of traffic (frames), each subject to the same flow metering and forwarding. Flow classification rules may be based on the following:

- a) Destination MAC address
- b) Source MAC address
- c) VID
- d) Priority

Item c), specifying a VID value, is not applicable to VLAN-unaware MAC Relays.

Frames classified using the same set of classification rules are subject to the same flow meter. The flow meter can change the drop_eligible parameter associated with each frame and can discard frames on the basis of the following parameters for each received frame and previously received frames, and the time elapsed since those frames were received:

- e) The received value of the drop_eligible parameter
- f) The mac_service_data_unit size

The flow meter shall not base its decision on the parameters of frames received on other Bridge Ports, or on any other parameters of those Ports. The metering algorithm described in the Metro Ethernet Forum (MEF) Technical Specification 10.3 (MEF 10.3) should be used.

NOTE 1—The flow meter described here can encompass a number of meters, each with a simpler specification. However, given the breadth of implementation choice permitted, further structuring to specify, for example, that frames can bypass a meter or are subject only to one of a number of meters provides no additional information.

NOTE 2—Although flow metering is applied after egress filtering (Figure 8-12), the meter(s) operate per reception Port, not per potential transmission Port(s).

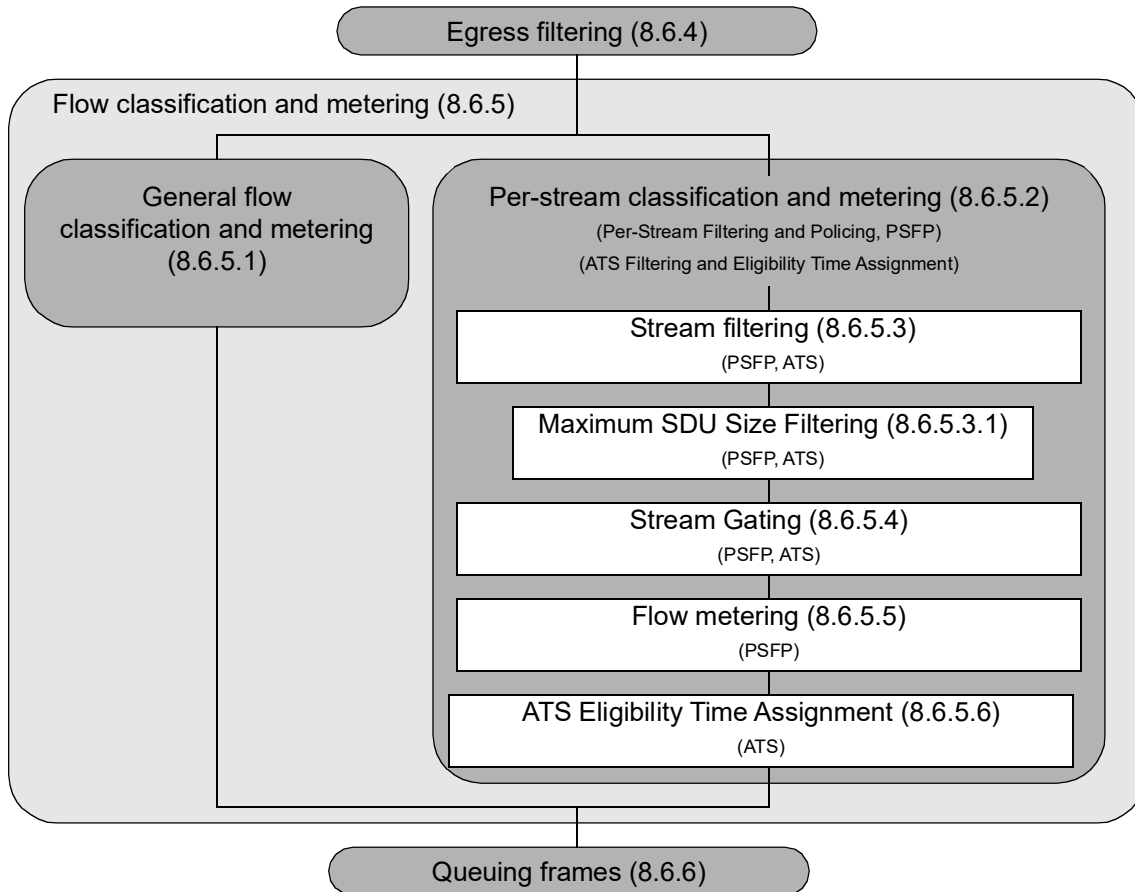


Figure 8-13—Flow classification and metering

8.6.5.2 Per-stream classification and metering

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

- Each received frame can be associated with a stream filter, as specified in 8.6.5.3. If a matching stream filter is specified (8.6.5.3), that is used to process the frame. Wildcard stream filters can be configured to match and discard frames not associated with a specified stream. If no matching stream filter is found, the frame is queued for transmission as specified in 8.6.6.
- If the stream filter specifies maximum SDU size filtering (8.6.5.3.1), that is used to process the frame. The frame can be discarded if a maximum SDU size is exceeded. The ATS scheduler state machine operation (8.6.11) assumes that the sizes of frames that it processes are less than or equal to the associated CommittedBurstSize parameter (8.6.11.3.5).
- The stream filter specifies a stream gate (8.6.5.4), that is used to process the frame. The frame can be discarded if it is received outside of permitted reception intervals or a given data limit within a reception interval is exceeded. The frame's priority can be mapped to an internal priority value (IPV) that can influence subsequent queuing decisions (8.6.6).
- If the stream filter specifies a flow meter (8.6.5.5), that is used to process the frame. The frame can be discarded or marked as drop eligible if a traffic limit of a flow meter is exceeded. A given stream filter can be configured with flow meters and an ATS scheduler if both PSFP and ATS are supported.

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

8.6.5.2.1 PSFP support

Each Bridge component or end station that implements PSFP shall support stream filtering, maximum SDU size filtering, stream gating, and flow metering, with the following:

- A single *Stream Filter Instance Table* (8.6.5.3).
- A single *Stream Gate Instance Table* (8.6.5.4).
- A single *Flow Meter Instance Table* (8.6.5.5).

The relationship between stream filters, stream gates, flow meters for streams subject to PSFP processing (as identified by the stream filter) is illustrated by Figure 8-14 for a number of streams.

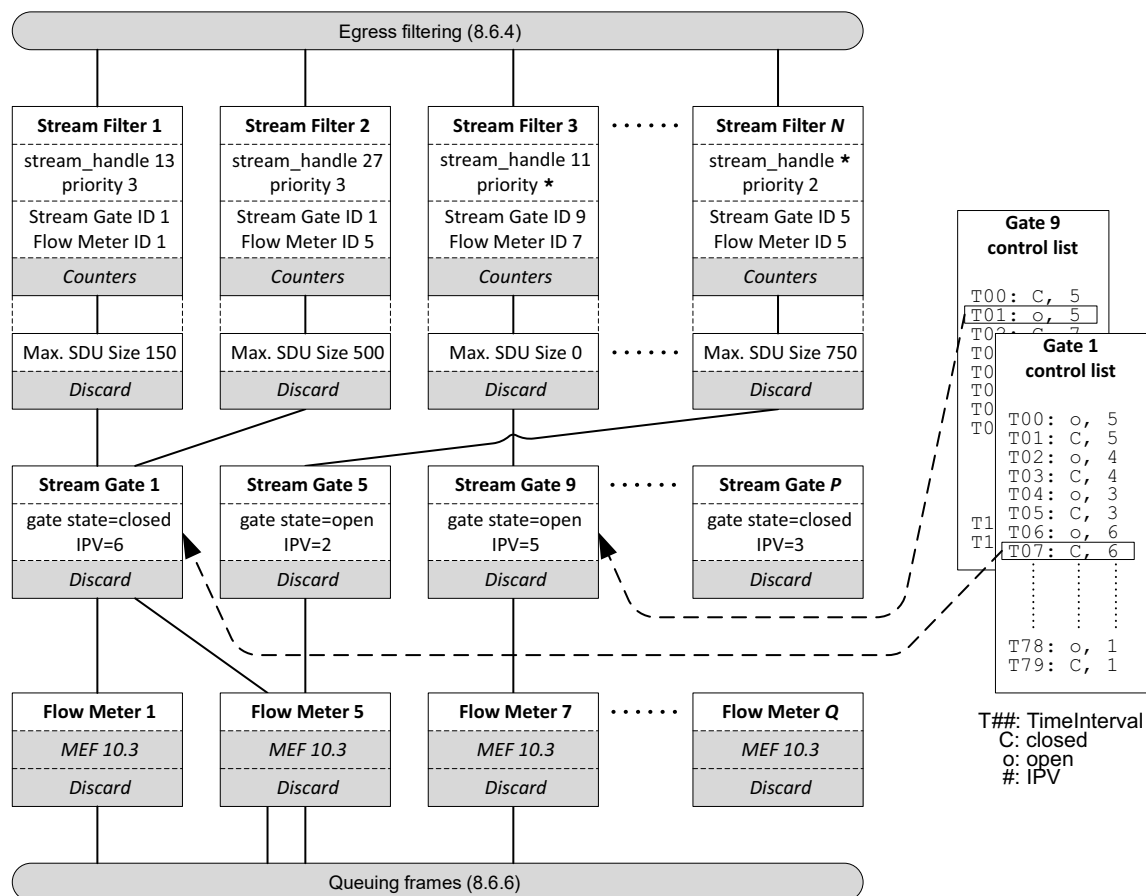


Figure 8-14—Per-stream classification for PSFP

8.6.5.2.2 ATS support

Each Bridge component that implements ATS shall support stream filtering, maximum SDU size filtering, stream gates supporting IPV assignment, ATS schedulers, and ATS scheduler groups, with the following:

- a) A single *Stream Filter Instance Table* (8.6.5.3).
- b) A single *Stream Gate Instance Table* (8.6.5.4).
If the Bridge component does not support PSFP in addition to ATS, each stream gate only supports IPV assignment (8.6.5.4). IPV can be used as part of adjusting per-hop delay bounds to meet specific networks' end-to-end delay requirements.
- c) A single *ATS Scheduler Instance Table* (8.6.5.6).
- d) A single *ATS Scheduler Group Instance Table* (8.6.5.6).
- e) An *ATS Port Parameter Table* for each Bridge Port (8.6.5.6).

NOTE 1—The operation of ATS schedulers (8.6.11) provides policing capabilities similar to those provided by flow meters (8.6.5.5). This handling of improper traffic can be sufficient, such that no additional flow meters are required.

NOTE 2—For bridges with support for ATS, and without support for PSFP, stream gates of ATS traffic will never close. In this case, stream gates are only used for IPV assignment.

NOTE 3—In presence of state changes of stream gates associated with ATS traffic, the asynchronous nature of ATS traffic can require that state changes of multiple of these associated gates are executed simultaneously.

The relationship between stream filters, stream gates, and ATS schedulers for streams subject to ATS processing is illustrated by Figure 8-15.

ATS support in end stations is provided by a modified variant of per-stream classification and metering for ATS, as specified in 49.1.

8.6.5.3 Stream filtering

A received frame can be associated with a stream filter using the frame's *stream_handle* and *priority* parameters. The *stream_handle* is a sub-parameter of the *connection_identifier* parameter of the ISS (6.6), provided by the stream identification function specified in Clause 6 of IEEE Std 802.1CB-2017 [B14].

Each stream filter comprises the following:

- a) An integer *stream filter identifier*.
- b) A *stream_handle* specification, either:
 - 1) A single value, as specified in IEEE Std 802.1CB.
 - 2) A wildcard, that matches any *stream_handle*.
- c) A *priority* specification, either:
 - 1) A single priority value.
 - 2) A wildcard value that matches any priority value.
- d) Maximum SDU size filtering (8.6.5.3.1) information, comprising:
 - 1) An integer *Maximum SDU size*, in octets. A value of 0 disables maximum SDU size filtering for this stream filter.
 - 2) A boolean *StreamBlockedDueToOversizeFrameEnable* parameter.
 - 3) A boolean *StreamBlockedDueToOversizeFrame* parameter.
- e) An integer *stream gate identifier* (8.6.5.4).
- f) An integer *flow meter instance identifier* (8.6.5.5).
If this parameter is absent, frames associated with the stream filter are not subject to flow metering.
- g) An integer *ATS scheduler instance identifier* (8.6.5.6).
If this parameter is absent, frames associated with the stream filter are not subject to ATS scheduling and transmission selection.

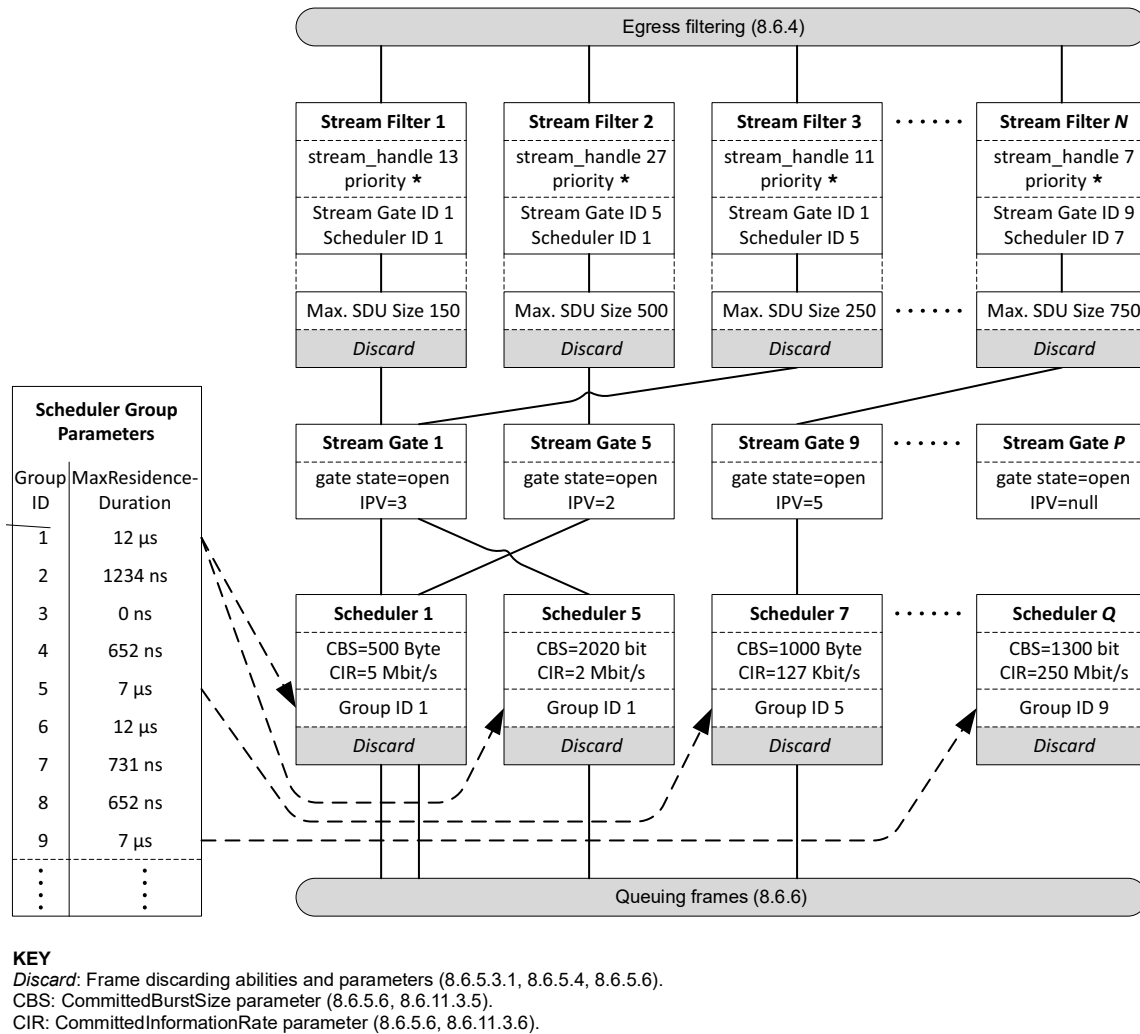


Figure 8-15—Per-stream classification and metering for ATS

Each stream filter also comprises the following counters for frames associated with the stream filter:

- h) *MatchingFramesCount*: all frames associated with that stream filter.
- i) *PassingSDUCount*: frames passing the maximum SDU size filter (8.6.5.3.1).
- j) *NotPassingSDUCount*: frames not passing the maximum SDU size filter (8.6.5.3.1).
- k) *PassingFrameCount*: frames passing the associated stream gate (8.6.5.4).
- l) *NotPassingFrameCount*: frames not passing the stream gate (8.6.5.4).
- m) *RedFramesCount*: frames discarded by the flow meter (8.6.5.5).

The *stream filter identifier* uniquely identifies the stream filter, indexing a *Stream Filter Instance Table* of up to *MaxStreamFilterInstances* stream filters. Each received frame is associated with the stream filter with the lowest *stream filter identifier* whose *stream_handle* and *priority* specification match the frame's parameters, and the *MatchingFramesCount* is incremented for that filter.

NOTE 1—The use of *stream_handle* and *priority*, along with the wild-carding rules previously stated, allow configuration possibilities that go beyond the selection of individual streams, for example, per-priority filtering and policing, or per-priority per-reception Port filtering and policing can be configured using these rules.

NOTE 2—If it is desired to discard frames that do not match any other stream filter, rather than such frames being processed without filtering, a stream filter can be placed at the end of the table with stream_handle and priority specifications both wild-carded and the stream gate identifier of a permanently closed stream gate.

NOTE 3—A discarded frame is not processed or counted by subsequent elements of the forwarding process. As a consequence, implementations can internally represent all counters [items h) through m) above] by the four counters MatchingFramesCount [item h) above], NotPassingSDUCount [item j) above], NotPassingFrameCount [item l) above], and RedFramesCount [item m) above].

8.6.5.3.1 Maximum SDU Size Filtering

If the SDU size of a frame exceeds the value of the associated stream filter's Maximum SDU size parameter, the frame is discarded and that stream filter's NotPassingSDUCount is incremented. If the stream filter's StreamBlockedDueToOversizeFrameEnable parameter is configured to be TRUE, the StreamBlockedDueToOversizeFrame parameter is set to TRUE and all subsequent frames will be discarded until StreamBlockedDueToOversizeFrame is administratively reset to FALSE.

Otherwise, the stream filter's PassingSDUCount is incremented (see 8.6.5.3). The default configuration of both StreamBlockedDueToOversizeFrameEnable and StreamBlockedDueToOversizeFrame is FALSE.

NOTE—The Maximum SDU size is defined per stream filter and can therefore differ from the queueMaxSDU specified in 8.6.8.4. As queueMaxSDU is applied after the flow classification and metering, it is possible that a frame that passes the Maximum SDU size filter will later be discarded because its SDU size exceeds queueMaxSDU.

8.6.5.4 Stream gating

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

NOTE 1—The IPV facilitates ATS per-hop delay bound adjustment to satisfy specific networks' end-to-end delay requirements. Annex (CQF) describes another IPV use case.

Each stream gate comprises the following:

- a) An integer *stream gate instance identifier*.
- b) An administrative and an operational *stream gate state* parameter. These parameters take the following values:
 - 1) Open: Frames are permitted to pass through the stream gate.
 - 2) Closed: Frames are not permitted to pass through the stream gate.
- c) An administrative and an operational *internal priority value specification* parameter. These parameters take the following values:
 - 1) Null, in this case the received frame's priority parameter is used as the IPV.
 - 2) A specific IPV for the frame.

The *stream gate instance identifier* uniquely identifies the stream gate, indexing a *Stream Gate Instance Table* of up to *MaxStreamGateInstances* stream gates.

If PSFP are supported, each stream gate also includes the following:

- d) An administrative and an operational *stream gate control list*.
- e) A boolean *GateClosedDueToInvalidRxEnable* parameter.
- f) A boolean *GateClosedDueToInvalidRx* parameter.
- g) A boolean *GateClosedDueToOctetsExceededEnable* parameter.
- h) A boolean *GateClosedDueToOctetsExceeded* parameter.

An instance of the stream gate control state machine (8.6.10) determines the operational values of the *stream gate state* and the *internal priority value specification* [items b) and c) above] by the cyclical execution of the control operations (see Table 8-4) specified in the stream gate's *stream gate control list* [item d) above]. The administrative *stream gate state* and *internal priority value specification* parameters are used to determine the initial values of the corresponding operational parameters, and the administrative *stream gate control list* parameter allows configuration of a new control list prior to enabling its use in a running system.

If a frame is passed by a stream gate, the *PassingFrameCount* of the stream filter (8.6.5.3) associated with that frame is incremented. The *NotPassingFrameCount* is incremented if the frame is discarded.

Table 8-4—Stream gate control operations

Operation name	Parameter(s)	Action
SetGateAndIPV	StreamGateState, IPV, TimeInterval, IntervalOctetMax	The StreamGateState parameter specifies a desired state, <i>open</i> or <i>closed</i> , for the stream gate, and the IPV parameter specifies a desired value of the IPV associated with the stream. On execution, the StreamGateState and IPV parameter values are used to set the operational values of the stream gate state and internal priority value specification parameters for the stream. After <i>TimeInterval</i> ticks (8.6.9.4.16) has elapsed since the completion of the previous stream gate control operation in the stream gate control list, control passes to the next stream gate control operation. The optional <i>IntervalOctetMax</i> parameter specifies the maximum number of MSDU octets that are permitted to pass the gate during the specified <i>TimeInterval</i> . If the <i>IntervalOctetMax</i> parameter is omitted, there is no limit on the number of octets that can pass the gate.

Stream gates are able to permanently discard frames and thus effectively override the operational gate state (i.e., the stream gate behaves as if the operational stream gate state is Closed). This capability is provided in case a frame is discarded due to a closed gate state [8.6.5.4 item b)], and in case a frame is discarded because there are insufficient octets left (i.e., the value of the *IntervalOctetsLeft* parameter, as specified in 8.6.10.8, is lower than the frame length).

- i) Permanent frame discarding due to a frame received during a closed gate state is enabled if the *GateClosedDueToInvalidRxEnable* parameter is TRUE, and disabled if this parameter is FALSE. If enabled and any frame is discarded, then the *GateClosedDueToInvalidRx* parameter is set to TRUE, and all subsequent frames are discarded as long as the *GateClosedDueToInvalidRxEnable* and *GateClosedDueToInvalidRx* parameters are TRUE. Changes of the *GateClosedDueToInvalidRxEnable* parameter and transitions from TRUE to FALSE of the *GateClosedDueToInvalidRx* parameter are administrative actions.
- j) Permanent frame discarding due to insufficient left octets is enabled if the *GateClosedDueToOctetsExceededEnable* parameter is TRUE, and disabled if this parameter is FALSE. If enabled and any frame is discarded because there are insufficient octets left, then the *GateClosedDueToOctetsExceeded* parameter is set to TRUE, and all subsequent frames are discarded as long as the *GateClosedDueToOctetsExceededEnable* and *GateClosedDueToOctetsExceededEnable* parameters are TRUE. Changes of the *GateClosedDueToOctetsExceededEnable* parameter and transitions from TRUE to FALSE of the *GateClosedDueToOctetsExceeded* parameter are administrative actions.

Per default, permanent frame discarding is disabled and all associated parameters have the default value FALSE.

NOTE 2—Permanent frame discarding allows the detection of incoming frames during time periods when the stream gate is in the closed state and exceptionally large ingress bursts to result in the stream gate behaving as it is in a

permanently closed state, until such a time as management action is taken to reset the condition. The intent is to support applications where the transmission and reception of frames across the network is coordinated such that frames are received only when the stream gate is open with a limited overall amount of ingress octets. Hence, frames received by the stream gate when it is in the closed state and unexpected amounts of ingress octets represent invalid receive conditions.

8.6.5.5 Flow metering

The flow meters specified by this subclause (8.6.5.5) implement the parameters and algorithm specified in *Bandwidth Profile Parameters and Algorithm* in MEF 10.3 with the additions described in this subclause.

Each flow meter comprises the following:

- a) An integer *flow meter identifier*.
- b) An integer *Committed information rate (CIR)*, in bits per second (MEF 10.3).
- c) An integer *Committed burst size (CBS)*, in octets (MEF 10.3).
- d) An integer *Excess Information Rate (EIR)*, in bits per second (MEF 10.3).
- e) An integer *Excess burst size (EBS) per bandwidth profile flow*, in octets (MEF 10.3).
- f) A *Coupling flag (CF)*, which takes the value 0 or 1 (MEF 10.3).
- g) A *Color mode (CM)*, which takes the value *color-blind* or *color-aware* (MEF 10.3).
- h) A boolean *DropOnYellow* parameter.
- i) A boolean *MarkAllFramesRedEnable* parameter.
- j) A boolean *MarkAllFramesRed* parameter.

NOTE 1—This standard specifies the relationship among frame length, media-dependent overhead, ATS scheduler operation, and the associated parameters in 8.6.11. In contrast, the operation and parameters of flow meters in 8.6.5.5 are as specified by MEF 10.3.

NOTE 2—Envelope and Rank, as defined in MEF 10.3, are not used by the flow meters described in this subclause; i.e., the reduced functionality algorithm described in 12.2 of MEF 10.3 is used.

The *flow meter identifier* uniquely identifies the flow meter instance, indexing a *Flow Meter Instance Table* of up to *MaxFlowMeterInstances* flow meters.

The *DropOnYellow* parameter indicates whether frames marked yellow by the MEF 10.3 algorithm are discarded or marked as drop eligible:

- k) A value of TRUE indicates that yellow frames are discarded.
- l) A value of FALSE indicates that the *drop_eligible* parameter of yellow frames is set to TRUE.

NOTE 3—Changing the value of the *drop_eligible* parameter may change the contents of the frame, depending on how the frame is tagged when transmitted, which may then require updating the *frame_check_sequence*. Mechanisms for conveying information from ingress to egress that the *frame_check_sequence* may require updating are implementation dependent.

Flow meters can permanently discard all frames after an initial frame has been discarded. This capability is enabled if the *MarkAllFramesRedEnable* parameter is TRUE, and disabled if this parameter is FALSE. If enabled and the flow meter discards a frame, then the *MarkAllFramesRed* parameter is set to TRUE, and all subsequent frames are discarded as long as the *MarkAllFramesRedEnable* and *MarkAllFramesRed* parameters are TRUE.

Changes of the *MarkAllFramesRedEnable* parameter and transitions from TRUE to FALSE of the *MarkAllFramesRed* parameter are administrative actions. Per default, permanent frame discarding is disabled and both associated parameters have the default value FALSE.

Each time a flow meter discards a frame, the *RedFramesCount* counter of the originating stream filter (8.6.5.3) is increased.

8.6.5.6 ATS eligibility time assignment

Asynchronous Traffic Shaping (ATS) schedulers assign eligibility times to frames which are then used for traffic regulation by the ATS transmission selection algorithm (8.6.8.5).

NOTE 1—ATS schedulers, as defined in this subclause (8.6.5.6), realize the computational part of the overall traffic shaping operation of ATS. The complete operation is provided in combination with the ATS transmission selection algorithm (8.6.8.5), which uses the assigned eligibility times to regulate the traffic for transmission.

Each ATS scheduler comprises the following:

- a) An integer *scheduler identifier*.
- b) An integer *scheduler group identifier*.
- c) An integer *CommittedBurstSizeParameter* parameter, in bits (8.6.11.3.5).
- d) An integer *CommittedInformationRate* parameter, in bits per second (8.6.11.3.6).
- e) An internal *bucket empty time* state variable, in seconds (8.6.11.3.3).

ATS schedulers are organized in *ATS scheduler groups*. There is one ATS scheduler group per reception Port per upstream traffic class, where the latter refers to the transmitting traffic class in the device connected to the given reception Port. All ATS schedulers that process frames from a particular reception Port and a particular upstream traffic class are in the respective ATS scheduler group.

Each ATS scheduler group comprises the following:

- f) An integer *scheduler group identifier*.
- g) An integer *MaximumResidenceTime* parameter, shared by all ATS schedulers in a scheduler group, in nanoseconds (8.6.11.3.13).
- h) An internal *group eligibility time* state variable, shared by all ATS schedulers in a scheduler group, in seconds (8.6.11.3.10).

NOTE 2—The organization of ATS schedulers into groups results in a non-decreasing ordering of eligibility times of successive frames associated with a single group. This permits frames of one group to be queued in FIFO order.

Each ATS scheduler assigns eligibility times to the associated frames, and discards frames in exceptional situations. The underlying operations are performed by an ATS scheduler state machine (8.6.11) associated with an ATS scheduler. This state machine updates the associated bucket empty time and group eligibility time state variables based on the *CommittedBurstSize*, the *CommittedInformationRate*, the *MaxResidenceTime*, the frame arrival times, and the frame lengths (including media-specific overhead).

Each Port is associated with the following variable for ATS schedulers:

- i) An integer *DiscardedFramesCount* of frames discarded by the ATS schedulers associated with that reception Port.

Each Bridge component provides an *ATS Scheduler Instance Table* of up to *MaxSchedulerInstances* ATS schedulers, an *ATS Scheduler Group Instance Table* of up to *MaxSchedulerGroupInstances* ATS scheduler groups, and a *ATS Scheduler Port Parameter Table* shared by all ATS schedulers associated with a reception Port.

NOTE 3—Whether ATS scheduler instances, ATS scheduler group instances, the scheduler instance table, and the scheduler group instance table are located in reception ports or in transmission ports is implementation specific.

8.6.6 Queuing frames

Change the fourth paragraph in 8.6.6 as shown (including splitting the paragraph into two paragraphs):

The Forwarding Process provides one or more queues for a given Bridge Port, each corresponding to a distinct traffic class. Each frame is mapped to a traffic class using the Traffic Class Table for the Port ~~and the frame's priority~~. The priority value used for this mapping is determined as follows:

- c) If stream gates are unsupported (8.6.5.4), the frame's priority is used.
- d) If stream gates are supported and the IPV specification assigned to the frame is the null value, the frame's priority is used.
- e) If stream gates are supported and the IPV specification assigned to the frame is an IPV, this IPV is used.

Traffic class tables may be managed. Table 8-5 shows the recommended mapping for the number of classes implemented, in implementations that do not support the credit-based shaper transmission selection algorithm (8.6.8.2). The requirements for priority to traffic class mappings in implementations that support the credit-based shaper transmission selection algorithm are defined in 34.5. Up to eight traffic classes may be supported, allowing separate queues for each priority.

Change NOTE 3 in 8.6.6 as shown:

NOTE 3—A queue in this context is not necessarily a single FIFO data structure. A queue is a record of all frames of a given traffic class awaiting transmission on a given Bridge Port. ~~The structure of this record is not specified. The transmission selection algorithm (8.6.8) determines which traffic class, among those classes with frames available for transmission, provides the next frame for transmission. The method of determining which frame within a traffic class is the next available frame is not specified beyond conforming to the frame ordering requirements of this subclause. This allows a~~ variety of queue structures such as a single FIFO, or a set of FIFOs with one for each ~~pairing of ingress and egress ports (i.e., Virtual Output Queuing), or a set of FIFOs with one for each~~ VLAN or priority, ~~or hierarchical structures is allowed.~~

Delete 8.6.6.1 ("PSFP queuing").

8.6.8 Transmission selection

Change the fourth paragraph of 8.6.8 as shown:

The strict priority transmission selection algorithm defined in 8.6.8.1 shall be supported by all Bridges as the default algorithm for selecting frames for transmission. The credit-based shaper transmission selection algorithm defined in 8.6.8.2, ~~and the ETS algorithm defined in 8.6.8.3,~~ and the ATS transmission selection algorithm defined in 8.6.8.5 may be supported in addition to the strict priority algorithm. Further transmission selection algorithms, selectable by management means, may be supported as an implementation option so long as the requirements of 8.6.6 are met.

Change Table 8-6 as shown:

Table 8-6—Transmission selection algorithm identifiers

Transmission selection algorithm	Identifier
Strict priority (8.6.8.1)	0
Credit-based shaper (8.6.8.2)	1
Enhanced Transmission Selection (ETS) (8.6.8.3)	2
ATS Transmission Selection (8.6.8.5)	3
Reserved for future standardization	~~34~~ –254
Vendor-specific Transmission Selection algorithm value for use with DCBX (D.2.8.8)	255
Vendor-specific	A four-octet integer, where the most significant 3 octets hold an OUI or CID value, and the least significant octet holds an integer value in the range 0–255 assigned by the owner of the OUI or CID.

8.6.8.4 Enhancements for scheduled traffic

Change the third paragraph of 8.6.8.4 as shown (including splitting the paragraph into two paragraphs and inserting a new Figure 8-17 and NOTE 1), and renumber the subsequent figures in Clause 8 accordingly:

In addition to the other checks carried out by the transmission selection algorithm, a frame on a traffic class queue is not available for transmission [as required for tests (a) and (b) in 8.6.8] if the transmission gate is in the closed state or if there is insufficient time available to transmit the entirety of that frame, [plus the media-dependent overhead specified in 12.4.2.2](#), before the next gate-close event (3.97) associated with that queue ([Figure 8-17](#)).

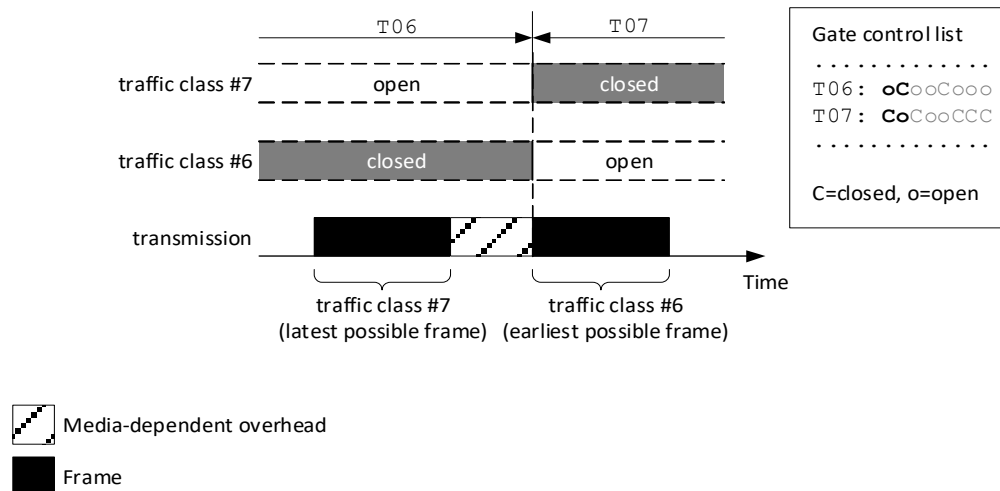


Figure 8-17—Frame timing at gate-close events

NOTE 1—For example, in the case of IEEE 802.3 media, the media-dependent overhead prior to the gate-close event includes the preamble of a potentially following frame from a different traffic class.

A per-traffic class counter, TransmissionOverrun (12.29.1.1.2), is incremented if the implementation detects that a frame from a given queue is still being transmitted by the MAC when the gate-close event for that queue occurs.

Change the former NOTE 1 of 8.6.8.4 as shown, and renumber the subsequent notes in this subclause accordingly:

~~NOTE 1—It is assumed that the implementation has knowledge of the transmission overheads that are involved in transmitting a frame on a given Port and can therefore determine how long the transmission of a frame will take. However, there can be reasons that the time required to complete the transmission of a frame is unknown. For example, where frame preemption is supported and there is no way of telling in advance how many times the given frame will be preempted before this transmission is complete, why the frame size, and therefore the length of time needed for its transmission, is unknown; for example, where cut through is supported, or where frame preemption is supported and there is no way of telling in advance how many times a given frame will be preempted before its transmission is complete.~~ It is desirable that the schedule for such traffic is designed to accommodate the intended pattern of transmission without overrunning the next gate-close event for the traffic classes concerned.

Insert the following subclause (8.6.8.5) after 8.6.8.4:

8.6.8.5 ATS transmission selection algorithm

For a given queue that supports ATS transmission selection, the algorithm determines that a frame is available for transmission if the queue contains one or more frames eligible for transmission. A frame is eligible for transmission if the assigned eligibility time (8.6.5.6, 8.6.11.3.2) is earlier than or at the current time.

The current time is determined by the TransmissionSelection Clock, which is an implementation specific local system clock function. The TransmissionSelection Clock determines the selectability time per frame, which is the time at which this frame is queued (8.6.6) and available for transmission selection. The selectability time is used as a reference to specify the handling of device-internal implementation specific timing properties (8.6.11.3.2).

All frames that reach their selectability time are selected for transmission in ascending order of the assigned eligibility times. Transmission selection of frames with identical assigned eligibility times shall maintain the ordering requirement specified in 8.6.6.

NOTE—In the case of frames with non-identical eligibility times, the ordering requirement from 8.6.6 is automatically satisfied due the operation of the ATS scheduler state machines (8.6.11), which assign eligibility times in a non-decreasing order.

8.6.10 Stream gate control state machines

Change the introductory text of 8.6.10 (including Table 8-8) as shown:

The execution of the gate operations in a stream gate control list (8.6.5.1.2, 8.6.5.4) is controlled by the three state machines specified in 8.6.9:

- a) The Cycle Timer state machine (8.6.9.1);
- b) The List Execute state machine (8.6.9.2); and
- c) The List Config state machine (8.6.9.3).

One instance of each state machine is instantiated for each stream gate control list associated with instances of stream gates in a Bridge component that supports **PSFPStream gates**. An overview of the operation of these state machines can be found in Figure 8-18.²

The operation of these state machines is as defined in 8.6.9, with the exception of the definitions of the ExecuteOperation() procedure, the SetGateStates() procedure, the ListPointer variable, the AdminGateStates variable, and the OperGateStates variable; amended versions of these definitions appear in 8.6.10.1 through 8.6.10.5. Table 8-8 shows the correspondence between the procedures/variables used in 8.6.9 and the **PSFPStream gate** versions of these procedures/variables.

Three additional variables needed by the Execute**PSFPStreamGate**Operation procedure are defined in 8.6.10.6 and 8.6.10.7.

Table 8-8—Scheduled Traffic and **PSFPStream Gate procedures/variables**

Procedure/variable name in 8.6.9	PSFPStream Gate procedure/variable name
ExecuteOperation() (8.6.9.2.1)	Execute PSFPStreamGate Operation() (8.6.10.1)
SetGateStates() (8.6.9.2.2)	Set PSFPStreamGate States() (8.6.10.2)
ListPointer (8.6.9.4.15)	PSFPStreamGate ListPointer (8.6.9.2.2)
AdminGateStates (8.6.9.4.5)	PSFPStreamGate AdminGateStates (8.6.10.4)
OperGateStates (8.6.9.4.22)	PSFPStreamGate OperGateStates (8.6.10.5)

Change 8.6.10.1 through 8.6.10.8 as shown:

8.6.10.1 Execute**PSFPStreamGate**Operation()

The Execute**PSFPStreamGate**Operation() procedure is responsible for fetching the next gate operation from the OperControlList, along with any parameters associated with it, and performing actions based upon the gate operation that has been fetched. The value of the **PSFPStreamGate**ListPointer variable (8.6.9.2.2) is used as an index into OperControlList. The procedure processes the operation according to its operation name (Table 8-4) as follows:

- If the operation name is SetGateAndIPV, then the StreamGateState parameter value associated with the operation is assigned to the **PSFPStreamGate**OperGateStates variable (8.6.10.5), the IPV parameter value is assigned to the OperIPV variable (8.6.10.7), and the TimeInterval parameter value associated with the operation is assigned to the TimeInterval variable (8.6.9.4.24). If the TimeInterval parameter value associated with the operation was 0, the TimeInterval variable is assigned the value 1. If there is an IntervalOctetMax parameter associated with the gate operation, then that parameter value is used to set the value of the IntervalOctetsLeft variable (8.6.10.8); otherwise, the IntervalOctetsLeft variable is set to a value greater than the maximum possible number of octets that the gate could pass during TimeInterval.
- If the operation name is unrecognized, then the **PSFPStreamGate**ListPointer variable (8.6.9.4.15) is assigned the value of the OperControlListLength variable (8.6.9.4.23) and the TimeInterval variable (8.6.9.4.24) is assigned the value 0.
- If there is no TimeInterval parameter associated with the operation, then the TimeInterval variable is assigned the value 0.

² Editor's note: Figure 8-15 (Scheduled traffic state machines—overview and relationships) as published in IEEE Std 802.1Q-2018 has been renumbered to Figure 8-18 because this amendment inserted new figures earlier in this clause.

8.6.10.2 SetPSFPStreamGateStates()

This procedure sets the stream gate state as specified by the value of the PSFPStreamGateOperGateStates variable (8.6.9.4.22).

8.6.10.3 PSFPStreamGateListPointer

An integer used as a pointer to entries in the OperControlList (8.6.9.4.19), each entry consisting of a stream gate control operation with its associated parameters (Table 8-4). A value of zero points at the first entry in the list; a value of (OperControlListLength – 1) points at the last entry.

8.6.10.4 PSFPStreamGateAdminGateStates

The initial state of the gate associated with the stream gate is set by the List Execute state machine (8.6.9.2) and is determined by the value of the PSFPStreamGateAdminGateStates variable. The default value of PSFPStreamGateAdminGateStates is open. The value of PSFPStreamGateAdminGateStates can be changed by management.

8.6.10.5 PSFPStreamGateOperGateStates

The current state of the gate associated with the stream gate. PSFPStreamGateOperGateStates is set by the List Execute state machine (8.6.9.2), and its initial value is determined by the value of the PSFPStreamGateAdminGateStates variable (8.6.10.4).

8.6.10.6 AdminIPV

The initial value of the OperIPV variable (8.6.10.7) associated with the stream gate is determined by the value of the AdminIPV variable. The default value of AdminIPV variable is the null value. The value of the AdminIPV variable can be changed by management.

8.6.10.7 OperIPV

The current value of the IPV associated with the stream gate. The initial value of OperIPV is set equal to the value of the AdminIPV variable (8.6.10.6). Subsequently, if there is a stream gate control list associated with the stream gate instance, its value is controlled by the contents of the operational stream gate control list and the operation of the List Execute state machine (8.6.9.2).

8.6.10.8 IntervalOctetsLeft

The current value of the IntervalOctetsLeft parameter variable indicates how many more MSDU octets can be passed by the stream gate during the current TimeInterval. This variable is initialized by the ExecutePSFPStreamGateOperation() procedure (8.6.10.1). If a frame that would otherwise pass the gate is larger than the current value of IntervalOctetsLeft, it is treated as if the gate is in the *closed* state; i.e., it is discarded. If a frame that would otherwise pass the gate is smaller than the current value of IntervalOctetsLeft, the number of MSDU octets is subtracted from the value of IntervalOctetsLeft.

Insert the following subclauses (8.6.11 through 8.6.11.3.14) after 8.6.10.8:

8.6.11 ATS Scheduler state machines

The ATS scheduler state machine operation is based on the ATS scheduler clocks (8.6.11.1), which is in relationship to transmission selection clocks (8.6.8.5). The state machine operation is specified by the ProcessFrame(frame) procedure (8.6.11.3) and the therein used state variables (8.6.11.3.3, 8.6.11.3.10). Parameters used by the ProcessFrame(frame) procedure are as defined in 8.6.5.6. Each arriving frame causes invocation of the ProcessFrame(frame) procedure.

The ATS scheduler state machine operation is based on a token bucket shaping algorithm as described in Tanenbaum and Wetherall [B86]. This subclause and its subclauses refer to the token bucket shaping algorithm, and the underlying terms, for informative explanation only. Specification of the ATS scheduler state machine operation does not depend on these explanations.

8.6.11.1 ATS Scheduler Clocks

An ATS scheduler clock is an implementation specific local system clock function. It is used to determine the arrival time of frames (8.6.11.3.1). A Bridge component may utilize one or more ATS scheduler clock instances. In the case of multiple scheduler clock instances, all ATS scheduler instances associated with the same reception Port share the same ATS scheduler clock instance (i.e., the arrival time of all frames received from a particular reception port is determined by the same ATS scheduler clock instance).

8.6.11.2 Relationship between ATS Scheduler Clocks and Transmission Selection Clocks

ATS scheduler clocks and transmission selection clocks (8.6.8.5) run at the same rate with bounded offset variation. A difference between an arbitrary instant of time t_{FA} , as recognized by an ATS scheduler clock instance, and the same instant of time t_{TS} , as recognized by a transmission selection clock instance, may be observed during the processing of a frame. The time difference may vary over a sequence of frames, which is characterized by the following equation:

$$\text{ClockOffsetMin} \leq t_{TS} - t_{FA} \leq \text{ClockOffsetMax}$$

where ClockOffsetMin and ClockOffsetMax are implementation specific constants that limit the variation. The range is characterized by ClockOffsetVariationMax (12.31.8.3) as follows:

$$\text{ClockOffsetVariationMax} = \text{ClockOffsetMax} - \text{ClockOffsetMin}$$

NOTE 1—ClockOffsetMin and ClockOffsetMax capture implementation specific properties such as the resolution of the associated clocks, associated rounding errors, constant offsets between clocks, Bridge-internal synchronization inaccuracies in presence of different underlying oscillators, and similar.

A pair of a scheduler clock instance and a transmission selection clock instance has an implementation specific nominal rate, and a maximal absolute deviation from this nominal rate during operation, as characterized by the ClockRateDeviationMax parameter (12.31.8.4).

NOTE 2—ClockRateDeviationMax captures implementation specific properties such as oscillator rate deviation, numeric resolution for the operations specified in 8.6.11.3, and similar.

NOTE 3—ATS scheduler clocks and transmission selection clocks provide a model to express different sources of delay, delay variation, and inaccuracy. It is not required to implement different multiple physical oscillators/clocks (i.e., ATS scheduler clocks and transmission selection clocks can actually be the same physical clock or can be generated from the same oscillator), but the model captures the properties of implementations with and without different physical oscillators/clocks in a unified manner.

8.6.11.3 ProcessFrame(frame)

This procedure computes eligibility time, assigns the eligibility times to frames, and updates the ATS scheduler state machine variables.

The procedure is described by the following pseudo-code in a neutral manner: The arithmetic precision, and the resolution of variables, are implementation specific, unless externally visible by management (12.31). The impact of the associated inaccuracies is discussed in Annex .

```
ProcessFrame(frame) {
    lengthRecoveryDuration    = length(frame) /
                               CommittedInformationRate;
    emptyToFullDuration       = CommittedBurstSize /
                               CommittedInformationRate;
    schedulerEligibilityTime  = BucketEmptyTime +
                               lengthRecoveryDuration;
    bucketFullTime            = BucketEmptyTime +
                               emptyToFullDuration;
    eligibilityTime            = max(arrivalTime(frame),
                                   GroupEligibilityTime,
                                   schedulerEligibilityTime);

    if (eligibilityTime <= (arrivalTime(frame) + MaxResidenceTime/1.0e9)) {
        // The frame is valid
        GroupEligibilityTime = eligibilityTime;
        BucketEmptyTime      = (eligibilityTime < bucketFullTime) ?
                               schedulerEligibilityTime :
                               schedulerEligibilityTime + eligibilityTime - bucketFullTime;
        AssignAndProceed(frame, eligibilityTime);
    } else {
        // The frame is invalid
        Discard(frame);
    }
}
```

8.6.11.3.1 arrivalTime(frame)

The arrival time of the frame, in seconds. The arrival time refers to the instant of time at which the associated ATS scheduler clock instance recognizes the arrival of the entire frame.

The point at which this recognition happens on the path between reception Ports and transmission Ports is implementation dependent. The earliest option is when the frame passes the boundary between the network physical medium and a reception Port, the latest option is during invocation of the ProcessFrame procedure.

For all frames arriving at all reception Ports, the arrival time is determined relative to the frame end. The maximum delay between the time a frame passes the boundary between the network physical medium and its subsequent recognition by the associated ATS scheduler clock instance is implementation specific, and is characterized by the ArrivalRecognitionDelayMax parameter (12.31.8.5).

The order of all frames associated with one ATS scheduler group at the boundary between the network physical medium and a reception Port is the same as the order of the arrival times determined for these frames.

NOTE—For example, the arrival time of frames may be determined based on invocation of the M_UNITDATA.indication service primitive of the ISS (6.6), as specified in IEEE Std 802.1AC.

8.6.11.3.2 AssignAndProceed(frame, eligibilityTime)

This procedure assigns an eligibility time to a frame for further processing by the transmission selection (8.6.8.5).

The assigned eligibility time, as used for ATS transmission selection decisions (8.6.8.5), is derived from the `eligibilityTime` parameter. The calculation of the assigned eligibility time accounts for variations between the associated ATS scheduler clock instance and transmission selection clock instance (8.6.11.2), and processing delays through the forwarding process, as detailed in the following.

Any frame may experience an additional, non-negative, processing delay, between its arrival time recognition by the associated ATS scheduler clock instance and its selectability time (8.6.8.5). This delay may vary per frame, thus that there is a delay variation over a sequence of frames. The processing delay is characterized by the following equation:

$$\text{ProcessingDelayMin} \leq \text{processingDelay}(\text{frame}) \leq \text{ProcessingDelayMax}$$

where `ProcessingDelayMin` (12.31.8.6) and `ProcessingDelayMax` (12.31.8.7) are implementation specific parameters, and `processingDelay(frame)` denotes the processing delay of an actual frame, including any potential delay introduced by the operation of the associated ATS scheduler state machine instance.

The assigned eligibility time is calculated as follows:

$$\text{assignedEligibilityTime} = \text{eligibilityTime} + \text{ClockOffsetMin} + \text{ProcessingDelayMax}$$

8.6.11.3.3 `BucketEmptyTime`

A state variable that contains the most recent instant of time at which the token bucket of the ATS scheduler instance was empty, in seconds.

The `BucketEmptyTime` variable is initialized with a time earlier than `CommittedBurstSize/CommittedInformationRate` in the past, as perceived by the ATS Scheduler Clock. After initialization, the number of tokens in the token bucket is equivalent to the `CommittedBurstSize` parameter.

8.6.11.3.4 `bucketFullTime`

The instant of time when the number of tokens in the token bucket is equivalent to the `CommittedBurstSize` parameter, in seconds.

8.6.11.3.5 `CommittedBurstSize`

The committed burst size of the ATS scheduler instance, in bits (8.6.5.6). The `CommittedBurstSize` parameter defines the maximum token capacity of the token bucket. In the token bucket model, the number of tokens removed from the bucket by a frame equals the length of the frame, as defined in 8.6.11.3.11.

8.6.11.3.6 `CommittedInformationRate`

The committed information rate of the ATS scheduler instance, in bits per second (8.6.5.6). The `CommittedInformationRate` parameter defines the rate at which the token bucket is refilled with tokens until the maximum token capacity of the token bucket is reached.

8.6.11.3.7 `Discard(frame)`

This procedure discards the frame and increases the `DiscardedFramesCount` counter of the associated reception port (8.6.5.6). The procedure is called in exceptional situations only (e.g., misbehavior of the connected upstream system).

8.6.11.3.8 eligibilityTime

The eligibility time of the frame, without taking the implementation specific device-internal timing properties of the forwarding process into account. These timing properties are taken into account by the AssignAndProceed(frame) procedure (8.6.11.3.2).

8.6.11.3.9 emptyToFullDuration

The duration required to accumulate a number of tokens equivalent to the CommittedBurstSize parameter, in seconds.

8.6.11.3.10 GroupEligibilityTime

A state variable that contains the most recent value of the eligibilityTime variable from the previous frame, as processed by any ATS scheduler instance in the same ATS scheduler group, in seconds.

The GroupEligibilityTime variable is initialized with a time earlier or equal to the current time, as perceived by the ATS scheduler clock.

8.6.11.3.11 length(frame)

The length of the frame, including all media-dependent overhead (12.4.2.2), in bits.

8.6.11.3.12 lengthRecoveryDuration

The duration required to accumulate a number of tokens equivalent to length(frame), in seconds.

8.6.11.3.13 MaxResidenceTime

The MaximumResidenceTime parameter of the ATS scheduler group instance associated with the ATS scheduler instance, in nanoseconds (8.6.5.6). The parameter limits the duration for which frames can reside in a Bridge.

NOTE—A consistent setup of MaxResidenceTime parameter can be determined by the per hop delay bound, which is a result of the timing analysis (Annex).

8.6.11.3.14 schedulerEligibilityTime

The instant of time when the number of tokens in the token bucket is at least equivalent to arrivalTime(frame), in seconds.

12. Bridge management

12.4 Bridge Management Entity

12.4.2 Port configuration

Change Table 12-2 as shown:

Table 12-2—Port table entry

Name	Data Type	Operations supported ^a	References
portComponentId	ComponentID	R	12.4.1.5
portPortNumber	Port Number	R	13.25
portMACAddress	MAC address	R	12.4.1.1.3 a)
portDelayExceededDiscards	counter	R	—
portMtuExceededDiscards	counter	R	—
portCapabilities	unsigned	R	—
portTypeCapabilities	unsigned	R	—
portType	enumerated	R	12.4.2.1
portExternal	Boolean	R	—
portAdminPointToPoint	unsigned	RW	IEEE Std 802.1AC
portOperPointToPoint	Boolean	R	IEEE Std 802.1AC
portName	Latin1 String (SIZE(0..32))	RW	—
portMediaDependentOverhead	unsigned	R	12.4.2.2

^a R = Read-only access; RW = Read/Write access.

Insert the following subclause (12.4.2.2) after 12.4.2.1:

12.4.2.2 Media-dependent overhead

The portMediaDependentOverhead parameter provides the number of additional octets for media-dependent framing. The overhead includes

- All octets prior to the first octet of the Destination Address field and
- The minimum number of octets after the last octet of the frame check sequence before a subsequent frame, including its media-dependent overhead, can be transmitted.

NOTE—An example of media-dependent overhead is 20 octets to account preamble, start of frame delimiter, and a minimal inter-frame gap (IFG) of 12 octets for IEEE 802.3 point-to-point media.

Change 12.31 as shown. This change includes subclause titles, introductory text, previously existing and new subclauses, and previously existing and new tables; see also the subsequent instructions for the insertion of 12.31.5 through 12.31.8.

12.31 Managed objects for per-stream ~~filtering~~classification and ~~policing~~metering

The Bridge enhancements for support of per-stream ~~filtering~~classification and ~~policing~~metering are defined in ~~8.6.5.1~~8.6.5.2, 8.6.5.3, 8.6.5.4, 8.6.5.5, and ~~the~~8.6.5.6. The associated state machines are defined in 8.6.10 and 8.6.11.

This managed resource comprises the following objects:

- a) The Stream Parameter Table (12.31.1)
- b) The Stream Filter Instance Table (12.31.2)
- c) The Stream Gate Instance Table (12.31.3)
- d) The Flow Meter Instance Table (12.31.4)
- e) The Scheduler Instance Table (12.31.5)
- f) The Scheduler Group Instance Table (12.31.6)
- g) The Scheduler Port Parameter Table (12.31.7)
- h) The Scheduler Timing Characteristics Table (12.31.8)

12.31.1 The Stream Parameter Table

There is one Stream Parameter Table per Bridge component. The table contains a set of parameters that supports PSFP (~~8.6.5.1~~8.6.5.2.1) and ATS (8.6.5.2.2), as detailed in Table 12-34.³ Tables can be created or removed dynamically in implementations that support dynamic configuration of Bridge components.

Table 12-34—The Stream Parameter Table

Name	Data type	Operations supported ^a	Conformance ^b	References
MaxStreamFilterInstances	integer	R	~~BE~~ <u>PSFP, ATS</u>	~~8.6.5.1~~ <u>8.6.5.3</u> , 12.31.2
MaxStreamGateInstances	integer	R	~~BE~~ <u>PSFP, ATS</u>	~~8.6.5.1~~ <u>8.6.5.4</u> , 12.31.3
MaxFlowMeterInstances	integer	R	~~BE~~ <u>PSFP, ats</u>	~~8.6.5.1~~ <u>8.6.5.5</u> , 12.31.4
SupportedListMax	integer	R	~~BE~~ <u>PSFP, ats</u>	~~8.6.5.1~~ <u>8.6.5.4</u> , 12.31.4
<u>MaxSchedulerInstances</u>	<u>integer</u>	<u>R</u>	<u>psfp, ATS</u>	<u>8.6.5.6, 12.31.5</u>
<u>MaxSchedulerGroupInstances</u>	<u>integer</u>	<u>R</u>	<u>psfp, ATS</u>	<u>8.6.5.6, 12.31.6</u>

^a R = Read only access; RW = Read/Write access.

^b ~~B~~ = Required for Bridge or Bridge component support of PSFP; ~~E~~ = Required for end station support of PSFP.
PSFP = Required for Bridge, Bridge component, or end station support of PSFP.
psfp = Optional for Bridge, Bridge component, or end station support of PSFP.
ATS = Required for Bridge or Bridge component support of ATS.
ats = Optional for Bridge or Bridge component support of ATS.

³ Editor's note: Table 12-31 through Table 12-34 as published in IEEE Std 802.1Q-2018 have been renumbered to Table 12-34 through Table 12-38, respectively, because IEEE Std 802.1Qcc-2018 inserted new tables earlier in this clause.

12.31.1.1 MaxStreamFilterInstances

The maximum number of Stream Filter instances supported by this Bridge component [\(8.6.5.3\)](#).

12.31.1.2 MaxStreamGateInstances

The maximum number of Stream Gate instances supported by this Bridge component [\(8.6.5.4\)](#).

12.31.1.3 MaxFlowMeterInstances

The maximum number of Flow Meter instances supported by this Bridge component [\(8.6.5.5\)](#).

12.31.1.4 SupportedListMax

The maximum value supported by this Bridge component of the AdminControlListLength [\(8.6.9.4.6\)](#) and OperControlListLength [\(8.6.9.4.23\)](#) parameters. It is available for use by schedule computation software to determine the Bridge component's control list capacity prior to computation.

12.31.1.5 MaxSchedulerInstances

[The maximum number of ATS scheduler instances supported by this Bridge component \(8.6.5.6\).](#)

12.31.1.6 MaxSchedulerGroupInstances

[The maximum number of ATS scheduler group instances supported by this Bridge component \(8.6.5.6\).](#)

12.31.2 The Stream Filter Instance Table

There is one Stream Filter Instance Table per Bridge component. Each table row contains a set of parameters that defines a single Stream Filter (~~8.6.5.1~~[8.6.5.3](#)), as detailed in Table 12-35. The table rows form an ordered list of filter instances, the order being determined by the StreamFilterInstance parameter. Tables can be created or removed dynamically in implementations that support dynamic configuration of Bridge components. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of stream filters.

Table 12-35—Stream Filter Instance Table

Name	Data type	Operations supported ^a	Conformance ^b	References
StreamFilterInstance	integer	R	~~BE~~ PSFP, ATS	~~8.6.5.1~~ 8.6.5.3
StreamHandleSpec	stream_handle specification	RW	~~BE~~ PSFP, ATS	~~8.6.5.1~~ 8.6.5.3
PrioritySpec	priority specification	RW	~~BE~~ PSFP, ATS	~~8.6.5.1~~ 8.6.5.3
MaximumSDUSize	integer	RW	PSFP, ATS	8.6.5.3.1 , 12.31.2.4
StreamGateInstanceID	integer	RW	~~BE~~ PSFP, ATS	~~8.6.5.1~~ 8.6.5.3 , ~~8.6.5.1.2~~ 8.6.5.4
~~FilterSpecificationList~~	~~sequence of FilterSpecification~~	~~RW~~	~~BE~~	~~8.6.5.1~~ , ~~8.6.5.1.3~~ , ~~12.31.2.6~~

Table 12-35—Stream Filter Instance Table (continued)

Name	Data type	Operations supported ^a	Conformance ^b	References
FlowMeterInstanceID	integer	RW	PSFP, ats	8.6.5.5, 12.31.2.6
FlowMeterEnable	Boolean	RW	PSFP, ats	8.6.5.5, 12.31.2.6
SchedulerInstanceID	integer	RW	psfp, ATS	8.6.5.6, 12.31.2.7
SchedulerEnable	Boolean	RW	psfp, ATS	8.6.5.6, 12.31.2.7
MatchingFramesCount	counter	R	~~B~~ E PSFP, ats	~~8.6.5.1~~ 8.6.5.3
PassingFramesCount	counter	R	~~B~~ E PSFP, ats	~~8.6.5.1~~ 8.6.5.3, 8.6.5.4
NotPassingFramesCount	counter	R	~~B~~ E PSFP, ats	~~8.6.5.1~~ 8.6.5.3, 8.6.5.4
PassingSDUCount	counter	R	~~B~~ E PSFP, ats	~~8.6.5.1~~ 8.6.5.3, 8.6.5.3.1
NotPassingSDUCount	counter	R	~~B~~ E PSFP, ats	~~8.6.5.1~~ 8.6.5.3, 8.6.5.3.1
REDFramesCount	counter	R	~~B~~ E PSFP, ats	~~8.6.5.1~~ 8.6.5.3, 8.6.5.5
StreamBlockedDueToOverSizeFrameEnable	Boolean	RW	~~B~~ E PSFP, ATS	~~8.6.5.1~~ 8.6.5.3, 8.6.5.3.1
StreamBlockedDueToOverSizeFrame	Boolean	RW	~~B~~ E PSFP, ATS	~~8.6.5.1~~ 8.6.5.3, 8.6.5.3.1

^a R = Read only access; RW = Read/Write access.

^b ~~B~~ = Required for Bridge or Bridge component support of PSFP; ~~E~~ = Required for end-station support of PSFP;
[PSFP](#) = Required for Bridge, Bridge component, or end station support of PSFP;
[psfp](#) = Optional for Bridge, Bridge component, or end station support of PSFP;
[ATS](#) = Required for Bridge or Bridge component support of ATS;
[ats](#) = Optional for Bridge or Bridge component support of ATS.

12.31.2.1 StreamFilterInstance

An integer index value that determines the place of the stream filter in the ordered list of stream filter instances. The values of StreamFilterInstance are ordered according to their integer value; smaller values appear earlier in the ordered list.

12.31.2.2 stream_handle specification data type

The stream_handle specification data type allows either of the following to be represented:

- A stream_handle value, represented as an integer.
- The wild card value.

12.31.2.3 priority specification data type

The priority specification data type allows either of the following to be represented:

- a) A priority value, represented as an integer.
- b) The wild card value.

12.31.2.4 MaxSDUSize

The MaxSDUSize parameter defines the maximum SDU size to be accepted by the stream filter (8.6.5.3). A value of 0 disables the maximum SDU size filtering for frames associated with the stream filter.

~~12.31.2.4 StreamGateInstance~~

12.31.2.5 StreamGateInstanceID

The StreamGateInstanceID parameter identifies the stream gate [instance](#) (12.31.3) that is associated with the stream filter. The relationship between stream filters and stream gates is many to one; a given stream filter can be associated with only one stream gate, but there can be multiple stream filters associated with a given stream gate.

~~12.31.2.5 Filter Specification data type~~

~~The FilterSpecification data type can represent the following:~~

- ~~a) An integer value representing a Maximum SDU size (8.6.5.1).~~
- ~~b) An integer value representing a flow meter instance identifier (8.6.5.1, 8.6.5.1.3).~~

12.31.2.6 FlowMeterInstanceID and FlowMeterEnable

If FlowMeterEnable is set to TRUE, the FlowMeterInstanceID parameter identifies the flow meter instance (12.31.4) that is associated with the stream filter. The relationship between stream filters and flow meters is many to one; a given stream filter can be associated with only one flow meter, but there can be multiple stream filters associated with a given flow meter. If FlowMeterEnable is set to FALSE, no flow meter instance is associated with the stream filter.

12.31.2.7 SchedulerInstanceID and SchedulerEnable

If SchedulerEnable is set to TRUE, the SchedulerInstanceID parameter identifies the ATS scheduler instance (12.31.5) that is associated with the stream filter. The relationship between stream filters and ATS schedulers is many to one; a given stream filter can be associated with only one ATS scheduler, but there can be multiple stream filters associated with a given ATS scheduler instance. If SchedulerEnable is set to FALSE, no ATS scheduler instance is associated with the stream filter.

12.31.3 The Stream Gate Instance Table

There is one Stream Gate Instance Table per Bridge component. Each table row contains a set of parameters that defines a single Stream Gate (~~8.5.6.1.28.6.5.4~~), as detailed in Table 12-36. Tables can be created or removed dynamically in implementations that support dynamic configuration of Bridge components. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of stream gates.

Table 12-36—The Stream Gate Instance Table

Name	Data type	Operations supported ^a	Conformance ^b	References
StreamGateInstance	integer	R	~~BE~~ PSFP, ATS	~~8.6.5.1,~~ ~~8.6.5.1.2~~ 8.6.5.4
~~PSFP~~ StreamGateEnabled	Boolean	RW	~~BE~~ PSFP, ATS	8.6.9.4.14
~~PSFP~~ StreamGateAdminGateStates	~~PSFP~~ StreamGateStatesV alue	RW	~~BE~~ PSFP, ATS	8.6.10.4, 12.29.1.2.2
~~PSFP~~ StreamGateOperGateStates	~~PSFP~~ StreamGateStatesV alue	R	~~BE~~ PSFP, ATS	8.6.10.5, 12.29.1.2.2
~~PSFP~~ StreamGateAdminControlListL ength	unsigned integer	RW	~~BE~~ PSFP, ats	8.6.9.4.6, 12.31.3.2
~~PSFP~~ StreamGateOperControlListLen gth	unsigned integer	R	~~BE~~ PSFP, ats	8.6.9.4.23, 12.31.3.2
~~PSFP~~ StreamGateAdminControlList	sequence of ~~PSFP~~ StreamGateControl Entry	RW	~~BE~~ PSFP, ats	8.6.9.4.2, 12.31.3.2, 12.31.3.2.2
~~PSFP~~ StreamGateOperControlList	sequence of ~~PSFP~~ StreamGateControl Entry	R	~~BE~~ PSFP, ats	8.6.9.4.19, 12.31.3.2, 12.31.3.2.2
~~PSFP~~ StreamGateAdminCycleTime	RationalNumber	RW	~~BE~~ PSFP, ats	8.6.9.4.3, 12.29.1.3
~~PSFP~~ StreamGateOperCycleTime	RationalNumber (seconds)	R	~~BE~~ PSFP, ats	8.6.9.4.20, 12.29.1.3
~~PSFP~~ StreamGateAdminCycleTimeEx tension	Integer (nanoseconds)	RW	~~BE~~ PSFP, ats	8.6.9.4.4
~~PSFP~~ StreamGateOperCycleTimeExte nsion	Integer (nanoseconds)	R	~~BE~~ PSFP, ats	8.6.9.4.21
~~PSFP~~ StreamGateAdminBaseTime	PTPtime	RW	~~BE~~ PSFP, ats	8.6.9.4.1, 12.29.1.4
~~PSFP~~ StreamGateOperBaseTime	PTPtime	R	~~BE~~ PSFP, ats	8.6.9.4.18, 12.29.1.4
~~PSFP~~ StreamGateConfigChange	Boolean	RW	~~BE~~ PSFP, ats	8.6.9.4.7
~~PSFP~~ StreamGateConfigChangeTime	PTPtime	R	~~BE~~ PSFP, ats	8.6.9.4.9, 12.29.1.4
~~PSFP~~ StreamGateTickGranularity	Integer (tenths of nanoseconds)	R	~~BE~~ PSFP, ats	8.6.9.4.16
~~PSFP~~ StreamGateCurrentTime	PTPtime	R	~~BE~~ PSFP, ats	8.6.9.4.10, 12.29.1.4
~~PSFP~~ StreamGateConfigPending	Boolean	R	~~BE~~ PSFP, ats	8.6.9.3, 8.6.9.4.8
~~PSFP~~ StreamGateConfigChangeError	Integer	R	~~BE~~ PSFP, ats	8.6.9.3.1

Table 12-36—The Stream Gate Instance Table (continued)

Name	Data type	Operations supported ^a	Conformance ^b	References
~~PSFP~~ StreamGateAdminIPV	IPV	RW	~~BE~~ PSFP, ATS	~~8.6.5.1.2~~ 8.6.5.4 , 8.6.10.6, 12.31.3.3
~~PSFP~~ StreamGateOperIPV	IPV	RW	~~BE~~ PSFP, ats	~~8.6.5.1.2~~ 8.6.5.4 , 8.6.10.7, 12.31.3.3
~~PSFP~~ StreamGateClosedDueToInvalidRxEnable	Boolean	RW	~~BE~~ PSFP, ats	~~8.6.5.1.2~~ 8.6.5.4
~~PSFP~~ StreamGateClosedDueToInvalidRx	Boolean	RW	~~BE~~ PSFP, ats	~~8.6.5.1.2~~ 8.6.5.4
~~PSFP~~ StreamGateClosedDueToOctetsExceededEnable	Boolean	RW	~~BE~~ PSFP, ats	~~8.6.5.1.2~~ 8.6.5.4
~~PSFP~~ StreamGateClosedDueToOctetsExceeded	Boolean	RW	~~BE~~ PSFP, ats	~~8.6.5.1.2~~ 8.6.5.4

^a R = Read only access; RW = Read/Write access.

^b ~~B~~—Required for Bridge or Bridge component support of PSFP; ~~E~~—Required for end-station support of PSFP.
[PSFP](#) = Required for Bridge, Bridge component, or end station support of PSFP.
[psfp](#) = Optional for Bridge, Bridge component, or end station support of PSFP.
[ATS](#) = Required for Bridge or Bridge component support of ATS.
[ats](#) = Optional for Bridge or Bridge component support of ATS.

12.31.3.1 StreamGateInstance

An integer table index that allows the stream gate to be referenced from Stream Filter Instance Table entries.

12.31.3.2 The gate control list structure and data types

The AdminControlList and OperControlList are ordered lists containing AdminControlListLength or OperControlListLength entries, respectively. Each entry represents a gate operation as defined in Table 8-4. Each entry in the list is structured as a GateControlEntry (12.31.3.2.2).

12.31.3.2.1 ~~PSFP~~[StreamGateStatesValue](#)

The ~~PSFP~~[StreamGateStatesValue](#) indicates the desired gate state, *open* or *closed*, for the stream gate.

12.31.3.2.2 ~~PSFP~~[StreamGateControlEntry](#)

A ~~PSFP~~[StreamGateControlEntry](#) consists of an operation name, followed by three parameters associated with the operation, as detailed in Table 8-4. The first parameter is a ~~PSFP~~[StreamGateStatesValue](#) (12.31.3.2.1); the second parameter is an IPV value (12.31.3.2.3), and the third parameter is a timeIntervalValue (12.31.3.2.4).

12.31.3.2.3 IPV value

The IPV value indicates the IPV (12.31.3.3) to be associated with frames that pass the gate (8.6.10.7).

12.31.3.2.4 timeIntervalValue

An unsigned integer, denoting a TimeInterval in nanoseconds (see TimeInterval in Table 8-4).

12.31.3.3 The Internal priority value (IPV) specification data type

The IPV data type represents an IPV value (~~8.6.5.1~~[8.6.5.4](#)); this is either the null value or an internal priority value.

12.31.3.4 Representation of times

Table 12-36 specifies times (e.g., ~~PSFPStream~~[GateAdminBaseTime](#)) with reference to the on-the-wire timing point at which the start of a frame crosses the boundary between the physical network media and PHY. This is the message timestamp point specified by IEEE Std 802.1AS for various media.

Replace Figure 12-7 with the following figure:

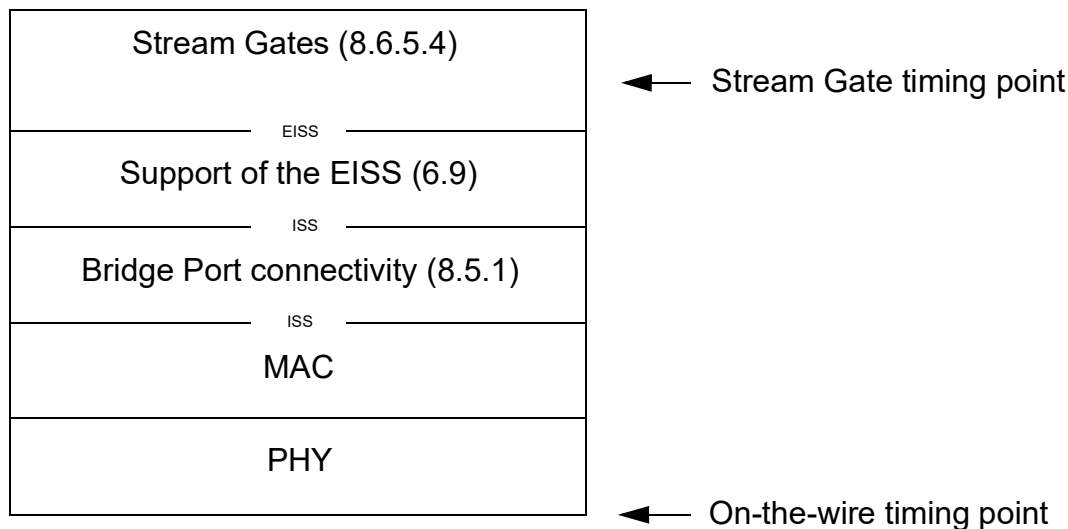


Figure 12-7—Timing points for Stream Gates

Figure 12-7 shows both the on-the-wire timing point and the ~~PSFPStream~~[Gate](#) timing point within the interface stack (above MAC and PHY) that is used for gate open/close as described in ~~8.6.5.1~~[8.6.5.4](#). Each timing point will have variance. A delay exists between the on-the-wire timing point and the ~~PSFPStream~~[Gate](#) timing point. This delay will have variance that is bounded (minimum/maximum).

The Bridge contains the information needed in order to compute the minimum/maximum delay from the on-the-wire timing point to the ~~PSFPStream~~[Gate](#) timing point. The Bridge shall adjust using the maximum delay, such that a frame's on-the-wire timing point occurs no later than the gate events represented in managed objects.

NOTE—Although the managed objects apply to a Bridge, the preceding specification of on-the-wire timing point can be applied to an end station.

12.31.4 The Flow Meter Instance Table

There is one Flow Meter Instance Table per Bridge component. Each table row contains a set of parameters that defines a single Flow Meter Instance (~~8.6.5.1~~[8.6.5.5](#)), as detailed in Table 12-37. Tables can be created or removed dynamically in implementations that support dynamic configuration of Bridge components. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of flow meters.

Table 12-37—The Flow Meter Instance Table

Name	Data type	Operations supported ^a	Conformance ^b	References
FlowMeterInstanceID	integer	R	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.2, 8.6.5.5
CIR	integer, bit/s	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5
CBS	integer, octets	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5
EIR	integer, bit/s	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5
EBS	integer, octets	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5
CF	integer, 0 or 1	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5
CM	enumerated, color-blind or color-aware	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5
DropOnYellow	Boolean	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5
MarkAllFramesRedEnable	Boolean	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5
MarkAllFramesRed	Boolean	RW	~~B~~ PSFP , ats	~~8.6.5.1, 8.6.5.1.3~~ 8.6.5.5

^a R = Read only access; RW = Read/Write access.

^b ~~B~~ = Required for Bridge or Bridge component support of PSFP; ~~E~~ = Required for end station support of PSFP.
[PSFP](#) = Required for Bridge, Bridge component, or end station support of PSFP.
[psfp](#) = Optional for Bridge, Bridge component, or end station support of PSFP.
[ATS](#) = Required for Bridge or Bridge component support of ATS.
[ats](#) = Optional for Bridge or Bridge component support of ATS.

Insert the following subclauses (12.31.5 through 12.31.8.7, including Table 12-38 through Table 12-41) after 12.31.4, and renumber the subsequent tables in Clause 12 accordingly:

12.31.5 The Scheduler Instance Table

There is one Scheduler Instance Table per Bridge component. Each table row in the Scheduler Instance Table comprises a set of parameters that defines a single ATS scheduler instance, as detailed in Table 12-38.

Table 12-38—The Scheduler Instance Table

Name	Data type	Operations supported ^a	Conformance ^b	References
SchedulerInstanceID	integer	R	psfp, ATS	8.6.5.6, 12.31.5.1
CommittedBurstSize	integer, bits	RW	psfp, ATS	8.6.5.6, 8.6.11.3.5, 12.31.5.2
CommittedInformationRate	integer, bits/s	RW	psfp, ATS	8.6.5.6, 8.6.11.3.6, 12.31.5.3
SchedulerGroupInstanceID	integer	RW	psfp, ATS	8.6.5.6, 12.31.5.4

^a R= Read only access; RW = Read/Write access.

^b PSFP = Required for Bridge, Bridge component, or end station support of PSFP.

psfp = Optional for Bridge, Bridge component, or end station support of PSFP.

ATS = Required for Bridge or Bridge component support of ATS.

ats = Optional for Bridge or Bridge component support of ATS.

NOTE—ATS scheduler groups establish the relationship between ATS scheduler instances and the per port management variables (12.31.7), as described in 8.6.5.6. As a result, the scheduler instance table does not contain references to ports.

12.31.5.1 SchedulerInstanceID

An integer table index that allows the ATS scheduler instance to be referenced from Stream Filter Instance Table entries.

12.31.5.2 CommittedBurstSize

As specified in 8.6.11.3.5.

12.31.5.3 CommittedInformationRate

As specified in 8.6.11.3.6.

12.31.5.4 SchedulerGroupInstanceID

The SchedulerGroupInstanceID parameter identifies the ATS scheduler group (12.31.6) that is associated with the ATS scheduler instance. Multiple scheduler instance can be associated to one ATS scheduler group, as detailed in 8.6.5.6.

12.31.6 The Scheduler Group Instance Table

There is one Scheduler Group Instance Table per Bridge component. Each table row in the Scheduler Group Instance Table comprises a set of parameters that defines a single ATS scheduler group instance (8.6.5.6), as detailed in Table 12-39.

Table 12-39—The Scheduler Group Instance Table

Name	Data type	Operations supported ^a	Conformance ^b	References
SchedulerGroupInstanceID	integer	R	psfp, ATS	8.6.5.6, 12.31.6.1
MaxResidenceTime	integer, nanoseconds	RW	psfp, ATS	8.6.5.6, 12.31.6.2

^a R = Read only access; RW = Read/Write access.

^b PSFP = Required for Bridge, Bridge component, or end station support of PSFP.

psfp = Optional for Bridge, Bridge component, or end station support of PSFP.

ATS = Required for Bridge or Bridge component support of ATS.

ats = Optional for Bridge or Bridge component support of ATS.

12.31.6.1 SchedulerGroupInstanceID

An integer table index that allows the ATS scheduler group instance to be referenced from Scheduler Instance Table entries.

12.31.6.2 MaxResidenceTime

As specified in 8.6.11.3.13.

12.31.7 The Scheduler Port Parameter Table

There is one Scheduler Port Parameter Table per Bridge. Each table row in the Scheduler Port Parameter Table comprises a set of parameters shared by all ATS scheduler instance associated with a reception Port, as detailed in Table 12-40.

Table 12-40—The Scheduler Port Parameter Table

Name	Data type	Operations supported ^a	Conformance ^b	References
PortNumber	integer	R	psfp, ATS	12.31.7.1
DiscardedFramesCount	integer	R	psfp, ATS	8.6.5.6, 8.6.11.3.7, 12.31.7.2

^a R = Read only access; RW = Read/Write access.

^b PSFP = Required for Bridge, Bridge component, or end station support of PSFP.

psfp = Optional for Bridge, Bridge component, or end station support of PSFP.

ATS = Required for Bridge or Bridge component support of ATS.

ats = Optional for Bridge or Bridge component support of ATS.