IEEE STANDARDS ASSOCIATION

IEEE Standard for
Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment 25: Enhancements for Scheduled Traffic

IEEE Computer Society

Sponsored by the
LAN/MAN Standards Committee

IEEE
3 Park Avenue
New York, NY 10016-5997
USA

IEEE Std 802.1Qbv™-2015
(Amendment to
IEEE Std 802.1Q™-2014
as amended by
IEEE Std 802.1Qca™-2015,
IEEE Std 802.1Qcd™-2015, and
IEEE Std 802.1Q-2014/Cor 1-2015)

IEEE Std 802.1Qbv™-2015
(Amendment to
IEEE Std 802.1Q™-2014
as amended by
IEEE Std 802.1Qca™-2015,
IEEE Std 802.1Qcd™-2015, and
IEEE Std 802.1Q-2014/Cor 1-2015)

IEEE Standard for
Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment 25: Enhancements for Scheduled Traffic

Sponsor

LAN/MAN Standards Committee
of the
IEEE Computer Society

Approved 5 December 2015

IEEE SA-Standards Board

Abstract: Enhancements to the forwarding process that supports scheduled traffic is provided in this amendment to IEEE Std 802.1Q-2014.

Keywords: Bridged Local Area Networks, IEEE 802®, IEEE 802.1Q™, IEEE 802.1Qca™, IEEE 802.1Qcd™, IEEE 802.1Qbv™, local area networks (LANs), MAC Bridges, metropolitan area networks, scheduled traffic, Virtual Bridged Local Area Networks (virtual LANs)

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

Print: ISBN 978-1-5044-0721-2 STD20807
PDF: ISBN 978-1-5044-0722-9 STDPD20807

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page, appear in all standards and may be found under the heading "Important Notice" or "Important Notices and Disclaimers Concerning IEEE Standards Documents."

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

IEEE Standards documents (standards, recommended practices, and guides), both full-use and trial-use, are developed within IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association ("IEEE-SA") Standards Board. IEEE ("the Institute") develops its standards through a consensus development process, approved by the American National Standards Institute ("ANSI"), which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute 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 does not warrant or represent the accuracy or content of the material contained in its standards, and expressly disclaims all warranties (express, implied and statutory) not included in this or any other document relating to the standard, including, but not limited to, the warranties of: merchantability; fitness for a particular purpose; non-infringement; and quality, accuracy, effectiveness, currency, or completeness of material. In addition, IEEE disclaims any and all conditions relating to: results; and workmanlike effort. 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: PROCUREMENT OF 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 should be considered 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, or 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 his or her views should be considered the personal views of that individual rather than the formal position of IEEE.

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE. However, IEEE does not provide 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 revisions to an IEEE standard is welcome to join the relevant IEEE working group.

Comments on standards should be submitted to the following address:

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

Laws and regulations

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

Copyrights

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

Photocopies

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

Updating of IEEE Standards documents

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

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

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE-SA Website at http://ieeexplore.ieee.org/xpl/standards.jsp or contact IEEE at the address listed previously. For more information about the IEEE–SA or IOWA's standards development process, visit the IEEE-SA Website at http://standards.ieee.org.

Errata

Errata, if any, for all IEEE standards can be accessed on the IEEE-SA Website at the following URL: http://standards.ieee.org/findstds/errata/index.html. Users are encouraged to check this URL for errata periodically.

Patents

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 http://standards.ieee.org/about/sasb/patcom/patents.html. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

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

Participants

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

Glenn Parsons, *Chair*
John Messenger, *Vice Chair*
Tony Jeffree, *Editor*
Michael Johas Teener, *Chair, Time Sensitive Networking Task Group*
Janos Farkas, *Vice Chair, Time Sensitive Networking Task Group*

Christian Boiger
Paul Bottorff
David Chen
Feng Chen
Weiying Cheng
Rodney Cummings
Norman Finn
Geoffrey Garner
Craig Gunther
Stephen Haddock
Mark Hantel
Marc Holness
Hal Keen
Stephan Kehrer
Marcel Kiessling
Philippe Klein
Jouni Korhonen
Yizhou Li
Christophe Mangin
Tom McBeath
James McIntosh
Hiroki Nakano
Bob Noseworthy
Donald R. Pannell
Walter Pienciak
Karen Randall
Maximilian Riegel
Dan Romascanu
Jessy Rouyer
Panagiotis Saltsidis
Michael Seaman
Daniel Sexton
Johannes Specht
Wilfried Steiner
Patricia Thaler
David Thornburg
Jeremy Touve
Paul Unbehagen
Karl Weber
Brian Weis
Jordon Woods
Helge Zinner
Juan Carlos Zuniga

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

Thomas Alexander
Butch Anton
Lee Armstrong
Christian Boiger
Ashley Butterworth
William Byrd
William Carney
Juan Carreon
Minho Cheong
Keith Chow
Charles Cook
Rodney Cummings
Patrick Diamond
Yezid Donoso
Sourav Dutta
Richard Edgar
Liu Fangfang
Janos Farkas
Hans-Joachim Fischer
Michael Fischer
Yukihiro Fujimoto
Devon Gayle
Gregory Gillooly
Eric W. Gray
David Gregson
Randall Groves
Craig Gunther
Stephen Haddock
Marco Hernandez
Werner Hoelzl
Noriyuki Ikeuchi
Sergiu Iordanescu
Atsushi Ito
Tony Jeffree
Michael Johas Teener
Peter Jones
Vincent Jones
Adri Jovin
Shinkyo Kaku
Piotr Karocki
Stuart Kerry
Yongbum Kim
Jouni Korhonen
Bruce Kraemer
Arthur H. Light
Elvis Maculuba
Arthur Marris
John Messenger
Charles Moorwood
Michael Newman
Nick S. A. Nikjoo
Satoshi Obara
Arumugam Paventhan
Alon Regev
Robert Robinson
Dan Romascanu
Jessy Rouyer
Bartien Sayogo
Michael Seaman
Shusaku Shimada
Veselin Skendzic
Kevin Stanton
Thomas Starai
Eugene Stoudenmire
Rene Struik
Walter Struppler
Mark-Rene Uchida
Dmitri Varsanofiev
Prabodh Varshney
George Vlantis
Khurram Waheed
Hung-Yu Wei
Andreas Wolf
Chun Yu Charles Wong
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 5 December 2015, it had the following membership:

John D. Kulick, *Chair*
Jon Walter Rosdahl, *Vice Chair*
Richard H. Hulett, *Past Chair*
Konstantinos Karachalios, *Secretary*

Masayuki Ariyoshi
Ted Burse
Stephen Dukes
Jean-Philippe Faure
J. Travis Griffith
Gary Hoffman
Michael Janezic
Joseph L. Koepfinger*
David J. Law
Hung Ling
Andrew Myles
T. W. Olsen
Glenn Parsons
Ronald C. Petersen
Annette D. Reilly
Stephen J. Shellhammer
Adrian P. Stephens
Yatin Trivedi
Phillip Winston
Don Wright
Yu Yuan
Daidi Zhong

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 802.1Qbv-2015, IEEE Standard for Local and metropolitan area networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic.

This amendment to IEEE Std 802.1Q-2014 provides enhancements to the forwarding process that support scheduled traffic.

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

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

Contents

Figures

Tables

IEEE Standard for
Local and metropolitan area networks—

Bridges and Bridged Networks

Amendment 25: Enhancements for Scheduled Traffic

(This amendment is based on IEEE Std 802.1Q™-2014 as amended by IEEE Std 802.1Qca™-2015, IEEE Std 802.1Qcd™-2015, and IEEE Std 802.1Q-2014/Cor 1-2015.)

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

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

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

2. Normative references

Insert the following reference in alphanumeric order:

IEEE Std 802.1AS™, IEEE Standard for Local and metropolitan area networks—Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks.

3. Definitions

Insert the following definitions in alphabetic order, renumbering as appropriate, and renumber the subsequent terms accordingly:

3.1 gate-close event: An event that occurs when the transmission gate associated with a queue transitions from the Open state to the Closed state, disconnecting the transmission selection function of the forwarding process from the queue and preventing it from selecting frames from that queue.

3.2 gate-open event: An event that occurs when the transmission gate associated with a queue transitions from the Closed state to the Open state, connecting the transmission selection function of the forwarding process to a queue and allowing it to select frames from that queue.

3.3 gating cycle: The period of time over which the sequence of operations in a gate control list repeats.

3.4 transmission gate: A gate that connects or disconnects the transmission selection function of the forwarding process from the queue, allowing or preventing it from selecting frames from that queue. The gate has two states, Open and Closed.

4. Abbreviations

Insert the following abbreviations in alphabetic order:

PTP IEEE 1588™ precision time protocol

TAI Temps Atomic International—International Atomic Time

5. Conformance

5.4 VLAN Bridge component requirements

5.4.1 VLAN Bridge component options

Insert the following items at the end of the lettered list, renumbering as necessary:

ac) Support the enhancements for scheduled traffic as specified in 8.6.8.4;
ad) Support the management entities for scheduled traffic as specified in 12.29.

5.13 MAC Bridge component requirements

5.13.1 MAC Bridge component options

Insert the following items at the end of the lettered list, renumbering as necessary:

l) Support the enhancements for scheduled traffic as specified in 8.6.8.4;
m) Support the state machines for scheduled traffic as specified in 8.6.9.

5.24 EVB station requirements

Insert the following new subclause after 5.24.1.2, at the end of Clause 5:

5.25 End station requirements—enhancements for scheduled traffic

An end station implementation that conforms to the provisions of this standard for enhancements for scheduled traffic shall:

a) Support the enhancements for scheduled traffic as specified in 8.6.8.4.
b) Support the state machines for scheduled traffic as specified in 8.6.9.

6. Support of the MAC Service

6.5 Quality of service (QoS) maintenance

6.5.2 Frame loss

Insert a new list item b8) at the end of the numbered list:

8) The Bridge supports enhancements for scheduled traffic (8.6.8.4) and the size of the service data unit exceeds the value of queueMaxSDU (8.6.8.4) for the traffic class queue on which the frame is to be queued.

8. Principles of bridge operation

8.6 The Forwarding Process

8.6.6 Queuing frames

Delete the fourth paragraph of 8.6.6 as shown:

~~VID is not applicable to VLAN-unaware MAC Relays and the above combinations will exclude any reference to VID when applied to MAC Bridges.~~

Change NOTE 2 of 8.6.6 as shown:

NOTE 2—Different ~~numbers of traffic classes may be implemented for different~~ Ports can use different numbers of traffic classes. Ports with media access methods that support a single transmission priority, such as Ethernet, can support more than one traffic class.

8.6.8 Transmission selection

Change the first paragraph of 8.6.8 as shown:

For each Port, frames are selected for transmission on the basis of the traffic classes that the Port supports and the operation of the transmission selection algorithms supported by the corresponding queues on that Port. For a given Port and ~~supported value of~~ traffic class, frames are selected from the corresponding queue for transmission if and only if

Insert a new paragraph at the end of 8.6.8 as shown:

When the enhancements for scheduled traffic are supported, additional requirements to determine whether a frame is available for transmission apply, as specified in 8.6.8.4.

8.6.8.2 Credit-based shaper algorithm

Change the text of 8.6.8.2 list item d), and add a NOTE, as shown:

d) ***idleSlope***. The rate of change of *credit*, in bits per second, when the value of *credit* is increasing (i.e., while *transmit* is FALSE and the transmission gate for the queue is open [8.6.8.4]). The value of *idleSlope* can never exceed *portTransmitRate*. If the enhancements for scheduled traffic (8.6.8.4) are not supported, or if GateEnabled is FALSE (8.6.9.4.14), ~~The~~ the value of *idleSlope* for a given queue is ~~determined by~~ equal to the value of the ***operIdleSlope(N)*** parameter for that queue, as defined in 34.3. If the enhancements for scheduled traffic (8.6.8.4) are supported, and GateEnabled is TRUE (8.6.9.4.14), then:

idleSlope = (operIdleSlope(N) × OperCycleTime / GateOpenTime)

where OperCycleTime is as defined in 8.6.9.4.20 and GateOpenTime is equal to the total amount of time during the gating cycle that the gate state for the queue is Open.

NOTE—When scheduled traffic operation is enabled, credit is accumulated only while the gate is open; therefore, the effective data rate of the idleSlope is increased to reflect the duty cycle for the transmission gate associated with the queue; however, the value of operIdleSlope(N) for the queue remains unchanged.

Change the text of 8.6.8.2 list item f) as shown:

f) ***credit.*** The transmission credit, in bits, that is currently available to the queue. If, at any time, there are no frames in the queue, and the *transmit* parameter is FALSE, and the transmission gate for the queue is open (8.6.8.4), and *credit* is positive, then *credit* is set to zero.

8.6.8.3 ETS algorithm

Insert a new subclause 8.6.8.4 after 8.6.8.3, renumbering as necessary:

8.6.8.4 Enhancements for scheduled traffic

A Bridge or an end station may support enhancements that allow transmission from each queue to be scheduled relative to a known timescale. In order to achieve this, a transmission gate is associated with each queue; the state of the transmission gate determines whether or not queued frames can be selected for transmission (see Figure 8-12). For a given queue, the transmission gate can be in one of two states:

a) *Open*: Queued frames are selected for transmission, in accordance with the definition of the transmission selection algorithm associated with the queue.
b) *Closed*: Queued frames are not selected for transmission.

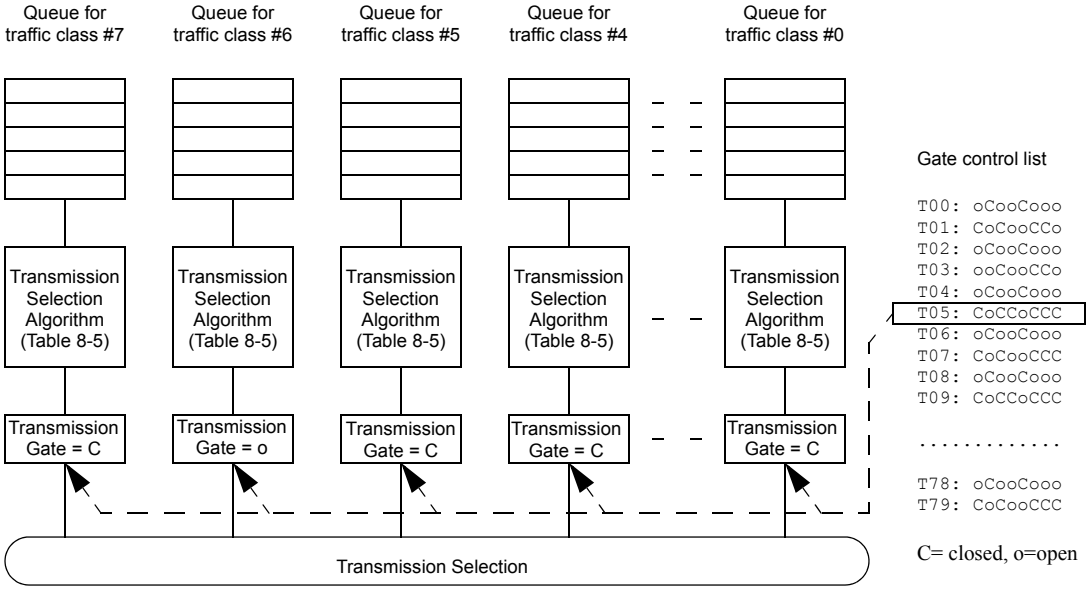


Figure 8-12—Transmission selection with gates

A *gate control list* associated with each Port contains an ordered list of gate operations. Each gate operation changes the transmission gate state for the gate associated with each of the Port's traffic class queues. In an implementation that does not support enhancements for scheduled traffic, all gates are assumed to be permanently in the *open* state. Table 8-6 identifies the gate operation types, their parameters, and the actions that result from their execution. The state machines that control the execution of the gate control list, along with their variables and procedures, are specified in 8.6.9.

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 before the next gate-close event (3.1) associated with that queue. 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.

Table 8-6—Gate operations

Operation name	Parameter(s)	Action
SetGateStates	GateState, TimeInterval	The *GateState* parameter indicates a value, *open* or *closed*, for each of the Port's queues. The gates are immediately set to the states indicated in the GateState parameter. This causes gate-close events (3.13.1) and/or gate-open events (3.23.2) to occur for any queue where the new GateState represents a change of state relative to the current state of the gate. After *TimeInterval* ticks (8.6.9.4.16) have elapsed since the completion of the previous gate operation in the gate control list, control passes to the next gate operation.

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 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.

NOTE 2— It is assumed that the implementation can determine the time at which the next gate-close event will occur from the sequence of gate events. In the normal case, this can be achieved by inspecting the sequence of gate operations in OperControlList (8.6.9.4.19) and associated variables. However, when a new configuration is about to be installed, it would be necessary to inspect the contents of both the OperControlList and the AdminControlList (8.6.9.4.2) and associated variables in order to determine the time of the next gate-close event, as the gating cycle for the new configuration may differ in size and phasing from the old cycle.

A per-traffic class queue queueMaxSDU parameter defines the maximum service data unit size for each queue; frames that exceed queueMaxSDU are discarded [6.5.2 b8)]. The value of queueMaxSDU for each queue is configurable by management (12.29); its default value is the maximum SDU size supported by the MAC procedures employed on the LAN to which the frame is to be relayed [6.5.2 b3)].

NOTE 3—The use of PFC is likely to interfere with a traffic schedule, because PFC is transmitted by a higher layer entity (see Clause 36).

In order for an end station to support the scheduled transmission of frames, it is necessary for its behavior to be compatible with the operation of the forwarding and queuing mechanisms employed in the Bridges to which it connects. In effect, the requirements for an end station are for its transmission selection to operate as if it is a single outbound Port of a Bridge that supports scheduled traffic. There are no particular requirements for end station support for the reception of scheduled traffic; only for the transmission of scheduled traffic.

Insert a new subclause 8.6.9 after 8.6.8, renumbering as necessary:

8.6.9 Scheduled traffic state machines

The execution of the gate operations in a Port's gate control list (8.6.8.4) is controlled by three state machines:

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 Port that supports the enhancements for scheduled traffic.

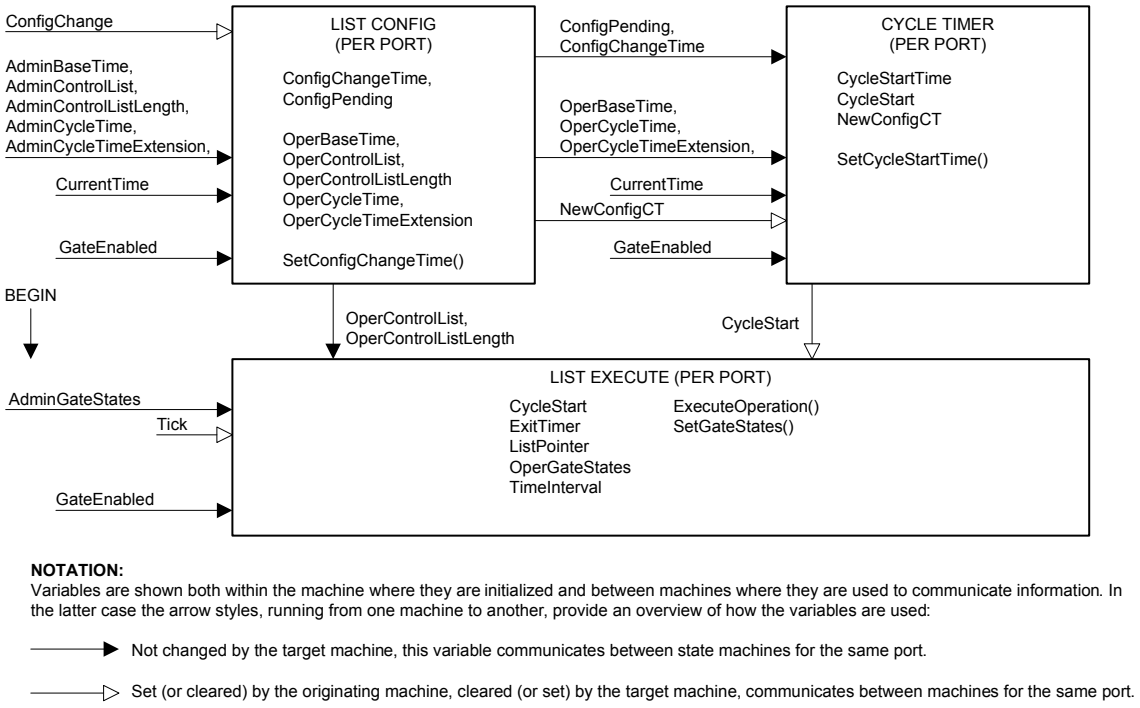
The Cycle Timer state machine initiates the execution of the gate control list and ensures that the gating cycle time defined for the Port is maintained.

The List Execute state machine executes the gate operations in the gate control list, in sequence, and establishes the appropriate time delay between each operation.

The List Config state machine manages the process of updating the current active schedule, interrupting the operation of the other two state machines while the update process is performed, and restarting them once the new schedule has been installed.

The state machine notation is specified in Annex E.

An overview of the state machines, showing their relationships and the variables that are used by them, can be seen in Figure 8-13.



NOTATION:
Variables are shown both within the machine where they are initialized and between machines where they are used to communicate information. In the latter case the arrow styles, running from one machine to another, provide an overview of how the variables are used:

→ Not changed by the target machine, this variable communicates between state machines for the same port.

⇢ Set (or cleared) by the originating machine, cleared (or set) by the target machine, communicates between machines for the same port.

Figure 8-13—Scheduled traffic state machines—overview and relationships

8.6.9.1 Cycle Timer state machine

The Cycle Timer state machine shall implement the function specified by the state diagram in Figure 8-14 and the attendant definitions in 8.6.9.1.1 and 8.6.9.4.

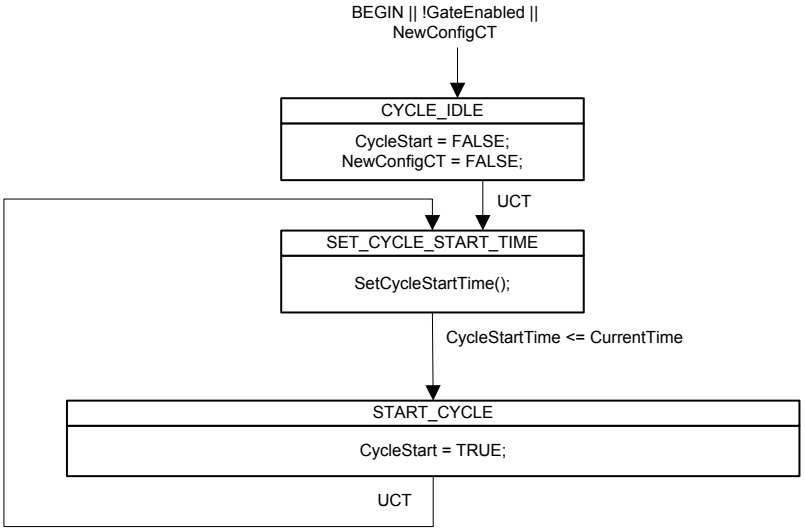


Figure 8-14—Cycle Timer state machine

8.6.9.1.1 SetCycleStartTime()

The SetCycleStartTime() procedure determines the time at which the next gate control list execution cycle is to start. The procedure sets the CycleStartTime variable (8.6.9.4.12) to the start time, based upon the values of CurrentTime (8.6.9.4.10), OperBaseTime (8.6.9.4.18), OperCycleTime (8.6.9.4.20), OperCycleTimeExtension (8.6.9.4.21), ConfigChangeTime (8.6.9.4.9), and ConfigPending (8.6.9.4.8), according to the following rules:

a) If:
ConfigPending = FALSE, and
OperBaseTime >= CurrentTime
(i.e., OperBaseTime specifies the current time or a future time)
Then:
CycleStartTime = OperBaseTime.

b) If:
ConfigPending = FALSE, and
OperBaseTime < CurrentTime
(i.e., OperBaseTime specifies a time in the past)
Then:
CycleStartTime = (OperBaseTime + N*OperCycleTime)
where N is the smallest integer for which the relation:
CycleStartTime >= CurrentTime
would be TRUE.

c) If:
ConfigPending = TRUE, and
ConfigChangeTime > (CurrentTime + OperCycleTime + OperCycleTimeExtension)
Then:
CycleStartTime = (OperBaseTime + N*OperCycleTime)
where N is the smallest integer for which the relation:

CycleStartTime >= CurrentTime
would be TRUE.

d) If:
ConfigPending = TRUE, and
ConfigChangeTime <= (CurrentTime + OperCycleTime + OperCycleTimeExtension)
Then:
CycleStartTime = ConfigChangeTime

NOTE 1—Since the origin of the PTP timescale is 1 January 1970 00:00:00 TAI, CycleStartTime will be larger than 1.3 x 10^18 ns. If sufficient precision is not maintained when computing N, CycleStartTime will not be an integer multiple of OperCycleTime, which could result in misalignment of the cycles at ports on different bridges.

NOTE 2—If AdminBaseTime is set to the same time in the past in all bridges and end stations, OperBaseTime is always in the past, and all cycles start synchronized. Using AdminBaseTime in the past is appropriate when you can start schedules prior to starting the application that uses the schedules. Use of AdminBaseTime in the future is intended to change a currently running schedule in all bridges and end stations to a new schedule at a future time. Using AdminBaseTime in the future is appropriate when schedules must be changed without stopping the application.

8.6.9.2 List Execute state machine

The List Execute state machine shall implement the function specified by the state diagram in Figure 8-15 and the attendant definitions in 8.6.9.2.1, 8.6.9.2.2, and 8.6.9.4.

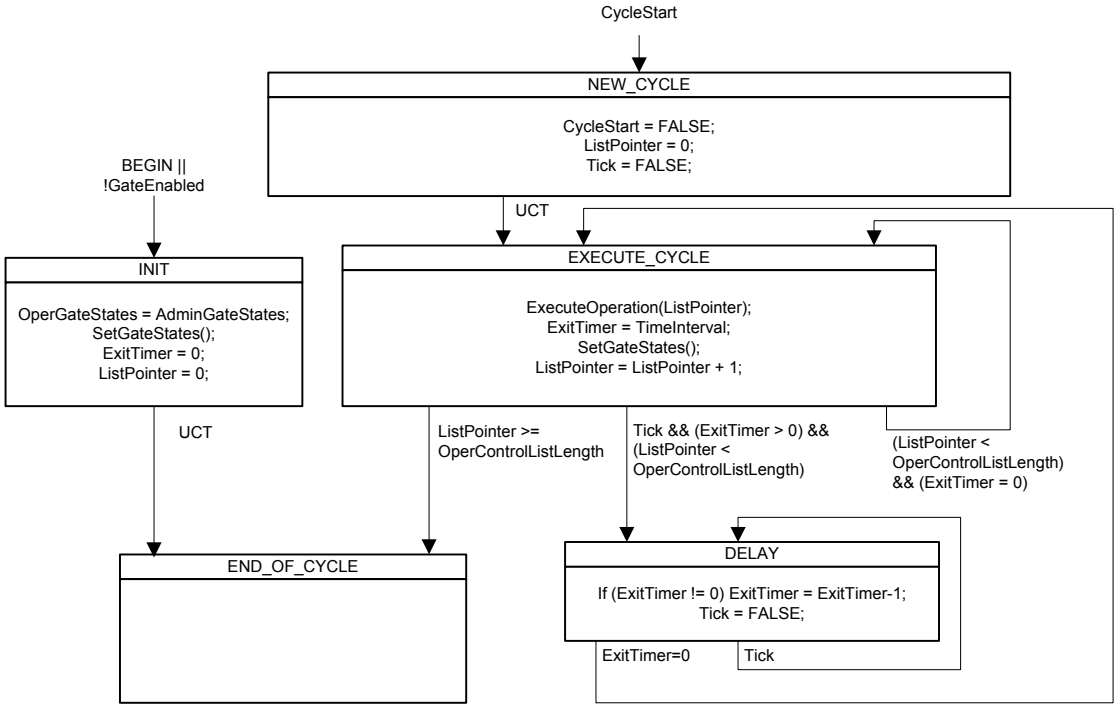


Figure 8-15—List Execute state machine

8.6.9.2.1 ExecuteOperation()

The ExecuteOperation() 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 ListPointer variable (8.6.9.4.15) is used as an index into OperControlList. The procedure processes the operation according to its operation name (Table 8-6) as follows:

a) If the operation name is SetGateStates, then the GateState parameter value associated with the operation is assigned to the OperGateStates variable (8.6.9.4.22), 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.

b) If the operation name is unrecognized, then the 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.

c) If there is no TimeInterval parameter associated with the operation, then the TimeInterval variable is assigned the value 0.

8.6.9.2.2 SetGateStates()

This procedure sets the gate state for each of the Port's queues as specified by the value of the OperGateStates variable (8.6.9.4.22).

NOTE—If the OperGateStates value differs from the previous gate states, the SetGateStates() procedure causes gate-open and/or gate-close events to occur (3.1, 3.2). It is possible that, on a given queue, the maximum time interval between any gate-open event and a subsequent gate-close event is smaller than the value of queueMaxSDU, in which case, frames that are too long to transmit but are shorter than queueMaxSDU could be queued on that queue. Such frames would never be transmitted, but would eventually be discarded because they exceed the maximum frame lifetime (6.5.2, 6.5.6). It should also be noted that a Bridge is allowed to discard frames that could never be transmitted [6.5.2 b8)].

8.6.9.3 List Config state machine

The List Config state machine shall implement the function specified by the state diagram in Figure 8-16 and the attendant definitions in 8.6.9.3.1 and 8.6.9.4.

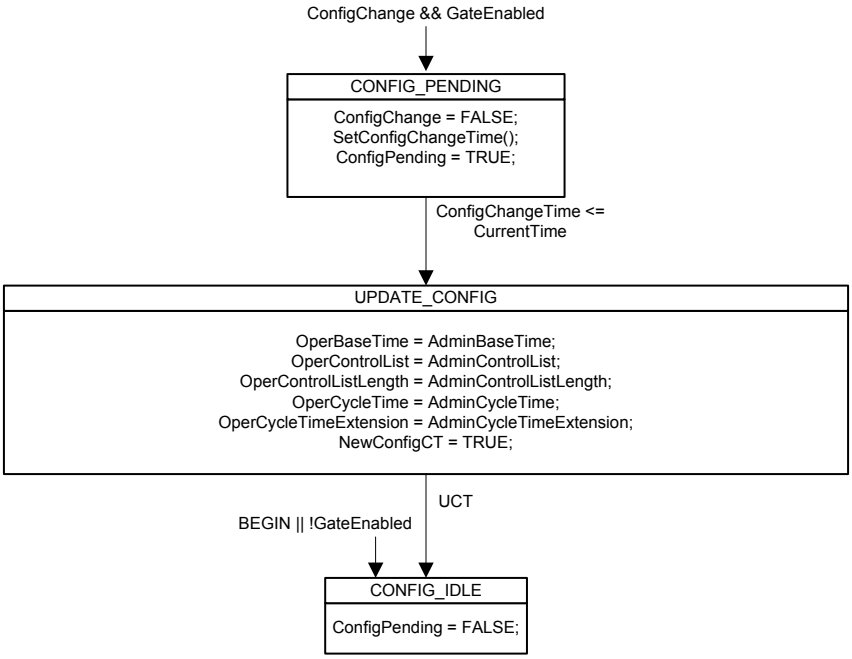


Figure 8-16—List Config state machine

8.6.9.3.1 SetConfigChangeTime()

The SetConfigChangeTime() procedure determines the time at which the administrative values of the cycle configuration variables are to be copied across to the operational variables. The procedure sets the ConfigChangeTime variable (8.6.9.4.9) to the start time, based upon the values of AdminBaseTime (8.6.9.4.1) and AdminCycleTime (8.6.9.4.3), according to the following rules:

a) If:
AdminBaseTime >= CurrentTime (8.6.9.4.10)
(i.e., AdminBaseTime specifies the current time or a future time)
Then:
ConfigChangeTime = AdminBaseTime

b) If:
(AdminBaseTime < CurrentTime) and (GateEnabled = FALSE)
(i.e., AdminBaseTime specifies a time in the past, and the current schedule is stopped)
Then:
ConfigChangeTime = (AdminBaseTime + N*AdminCycleTime)
where N is the smallest integer for which the relation
ConfigChangeTime >= CurrentTime
would be TRUE.

c) If:
(AdminBaseTime < CurrentTime) and (GateEnabled = TRUE)
(i.e., AdminBaseTime specifies a time in the past, and the current schedule is running)
Then:
Increment ConfigChangeError counter (12.29.1)
ConfigChangeTime = (AdminBaseTime + N*AdminCycleTime)
where N is the smallest integer for which the relation
ConfigChangeTime >= CurrentTime
would be TRUE.

8.6.9.4 State machine variables

8.6.9.4.1 AdminBaseTime

The administrative value of base time, expressed as an IEEE 1588 precision time protocol (PTP) timescale (see 8.2 of IEEE Std 802.1AS-2011). This value can be changed by management, and is used by the List Config state machine (8.6.9.3) to set the value of OperBaseTime (8.6.9.4.18).

NOTE—Time is expressed in the PTP timescale as the number of seconds, nanoseconds, and fractional nanoseconds that have elapsed since 1 January 1970 00:00:00 TAI.

8.6.9.4.2 AdminControlList

The administrative version of the gate control list for the Port. This value can be changed by management, and is used by the List Config state machine (8.6.9.3) to set the value of OperControlList (8.6.9.4.19).

8.6.9.4.3 AdminCycleTime

The administrative value of the gating cycle for the Port (3.3). This value can be changed by management, and is used by the List Config state machine (8.6.9.3) to set the value of OperCycleTime (8.6.9.4.20). The AdminCycleTime variable is a rational number of seconds, defined by an integer numerator and an integer denominator.

8.6.9.4.4 AdminCycleTimeExtension

An integer number of nanoseconds, defining the maximum amount of time by which the gating cycle for the Port (3.3) is permitted to be extended when a new cycle configuration is being installed. This administrative value can be changed by management, and is used by the List Config state machine (8.6.9.3) to set the value of OperCycleTimeExtension (8.6.9.4.21).

8.6.9.4.5 AdminGateStates

The initial state of the gate associated with each queue for the Port is set by the List Execute state machine (8.6.9.2), and is determined by the value of the AdminGateStates variable. The default value of AdminGateStates is *open* for all queues. The value of AdminGateStates can be changed by management.

8.6.9.4.6 AdminControlListLength

This value can be changed by management, and is used by the List Config state machine (8.6.9.3) to set the value of OperControlListLength (8.6.9.4.23).

8.6.9.4.7 ConfigChange

A Boolean variable that acts as a start signal to the List Config state machine (8.6.9.3) that the administrative variable values for the Port are ready to be copied into their corresponding operational variables. This variable is set TRUE by management and is set FALSE by the List Config state machine.

8.6.9.4.8 ConfigPending

A Boolean variable, set TRUE by the List Config state machine (8.6.9.3) to signal that there is a new cycle configuration awaiting installation. The variable is set FALSE when the List Config state machine has installed the new configuration. The variable is used by the SetCycleStartTime() procedure (8.6.9.1.1) to control the length of the cycle that immediately precedes the first cycle that uses the new configuration values. This value can be read by management, as specified in 12.29.1.

8.6.9.4.9 ConfigChangeTime

The time at which the administrative variables that determine the cycle are to be copied across to the corresponding operational variables, expressed as a PTP timescale. The value of this variable is set by the SetConfigChangeTime() procedure (8.6.9.3.1) in the List Config state machine (8.6.9.3).

8.6.9.4.10 CurrentTime

The current time maintained by the local system, expressed as a PTP timescale (see 8.2 of IEEE Std 802.1AS-2011).

NOTE 1—Other sources of times can be used to provide a value for the time. The implementation has to ensure that the same view of time is shared by the implementation and whatever entity is responsible for establishing the time schedule. The described mechanisms will work also with other formats of time.

NOTE 2—A discontinuity in time will have a predictable effect upon the operation of the state machines, as a consequence of their definitions. The implementer and/or user is advised to understand what those effects could be, and either limit such discontinuities or ensure that the results will not adversely affect the application.

8.6.9.4.11 CycleStart

A Boolean variable used as a signal from the Cycle Timer state machine (8.6.9.1) to the List Execute state machine to start executing the gate control list. Set TRUE by the Cycle Timer state machine and set FALSE by the List Execute state machine (8.6.9.2).

8.6.9.4.12 CycleStartTime

The time at which the next gate control list execution cycle is to start, expressed as a PTP timescale. The value of this parameter is set by the SetCycleStartTime() procedure (8.6.9.1.1) in the Cycle Timer state machine (8.6.9.1).

8.6.9.4.13 ExitTimer

A timer that implements the delay associated with the currently executing gate operation, expressed as an integer number of nanoseconds. The value is set by the operation of the List Execute state machine (8.6.9.2).

8.6.9.4.14 GateEnabled

A Boolean variable that indicates whether the operation of the state machines is enabled (TRUE) or disabled (FALSE). This variable is set by management. The default value of this variable is FALSE.

8.6.9.4.15 ListPointer

An integer used as a pointer to entries in the OperControlList (8.6.9.4.19), each entry consisting of a gate operation with its associated parameters (Table 8-6). A value of zero points at the first entry in the list; a value of (OperControlListLength)–1 points at the last entry.

8.6.9.4.16 Tick

A Boolean variable, set to TRUE by an implementation-specific system clock function at one nanosecond intervals, that controls the decrementing of the ExitTimer variable (8.6.9.4.13). This variable is set FALSE by the operation of the List Execute state machine (8.6.9.2).

NOTE—While the state machine is documented on the basis of a nanosecond clock “tick,” it is anticipated that real implementations will use a wide variety of clocks that differ in frequency accuracy and granularity. Hence, the management parameters specified in 12.29 allow a management station to discover the characteristics of an implementation’s cycle timer clock (TickGranularity) and to set the parameters for the gating cycle accordingly.

8.6.9.4.17 NewConfigCT

A Boolean variable that is used as a signal from the List Config state machine (8.6.9.3) to the Cycle Timer state machine (8.6.9.1) to indicate that a new configuration is being processed. The variable is set TRUE by the operation of the List Config state machine and set FALSE by the operation of the Cycle Timer state machine (8.6.9.1).

8.6.9.4.18 OperBaseTime

The operational value of base time, expressed as a PTP timescale (see 8.2 of IEEE Std 802.1AS-2011). This variable is used by the List Config state machine (8.6.9.3).

8.6.9.4.19 OperControlList

The active Gate Control List for the Port; the List Execute state machine executes this list. The contents of the list is populated dynamically from the AdminControlList (8.6.9.4.2) under the control of the List Config state machine (8.6.9.3).

8.6.9.4.20 OperCycleTime

The operational value of the gating cycle for the Port (3.3). This variable is set dynamically from the AdminCycleTime variable (8.6.9.4.3) under the control of the List Config state machine (8.6.9.3). OperCycleTime is used by the Cycle Timer state machine (8.6.9.1) to enforce the cycle time for the Port. The OperCycleTime variable is a rational number of seconds, defined by an integer numerator and an integer denominator.

8.6.9.4.21 OperCycleTimeExtension

An integer number of nanoseconds, defining the maximum amount of time by which the gating cycle for the Port (3.3) is permitted to be extended when a new cycle configuration is installed. This operational value is set by the List Config state machine (8.6.9.3) to the value of AdminCycleTimeExtension (8.6.9.4.4). The value of OperCycleTimeExtension is used by the SetCycleStartTime() procedure (8.6.9.1.1).

8.6.9.4.22 OperGateStates

The current state of the gate associated with each queue for the Port. OperGateStates is set by the List Execute state machine (8.6.9.2), and its initial value is determined by the value of the AdminGateStates variable (8.6.9.4.5).

8.6.9.4.23 OperControlListLength

This variable is set dynamically from the AdminControlListLength parameter (8.6.9.4.6) under the control of the List Config state machine (8.6.9.3). OperControlListLength indicates the number of entries in the OperControlList and is used by the List Execute state machine (8.6.9.2) to determine when the end of the list has been reached.

8.6.9.4.24 TimeInterval

Set to the value of the TimeInterval parameter associated with the currently executing gate operation. Used in the operation of the List Execute state machine (8.6.9.2).

12. Bridge management

Insert a new subclause 12.29 after 12.28, as shown, renumbering as necessary.

12.29 Managed objects for scheduled traffic

The Bridge enhancements for support of scheduled traffic are defined in 8.6.8 and 8.6.8.4.

The objects that comprise this managed resource are as follows:

a) The Gate Parameter Table (12.29.1)

12.29.1 The Gate Parameter Table

There is one Gate Parameter Table per Port of a Bridge component. Each table row contains a set of parameters that supports the enhancements for scheduled traffic (8.6.8.4), as detailed in Table 12-28. Rows in the table can be created or removed dynamically in implementations that support dynamic configuration of ports and components.

Table 12-28—The Gate Parameter Table

Name	Data type	Operations supported[a]	Conformance[b]	References
queueMaxSDUTable[c]	sequence of queueMaxSDU	RW	BE	8.6.8.4, 12.29.1.1, 12.29.1.1.1
GateEnabled	Boolean	RW	BE	8.6.9.4.14
AdminGateStates	gateStatesValue	RW	BE	8.6.9.4.5, 12.29.1.2, 12.29.1.2.2
OperGateStates	gateStatesValue	R	BE	8.6.9.4.22, 12.29.1.2, 12.29.1.2.2
AdminControlListLength	unsigned integer	RW	BE	8.6.9.4.6, 12.29.1.2
OperControlListLength	unsigned integer	R	BE	8.6.9.4.23, 12.29.1.2
AdminControlList	sequence of GateControlEntry	RW	BE	8.6.9.4.2, 12.29.1.2, 12.29.1.2.1
OperControlList	sequence of GateControlEntry	R	BE	8.6.9.4.19, 12.29.1.2, 12.29.1.2.1
AdminCycleTime	RationalNumber	RW	BE	8.6.9.4.3, 12.29.1.3
OperCycleTime	RationalNumber (seconds)	R	BE	8.6.9.4.20, 12.29.1.3
AdminCycleTimeExtension	Integer (nanoseconds)	RW	BE	8.6.9.4.4

Table 12-28—The Gate Parameter Table *(continued)*

Name	Data type	Operations supported[a]	Conformance[b]	References
OperCycleTimeExtension	Integer (nanoseconds)	R	BE	8.6.9.4.21
AdminBaseTime	PTPtime	RW	BE	8.6.9.4.1, 12.29.1.4
OperBaseTime	PTPtime	R	BE	8.6.9.4.18, 12.29.1.4
ConfigChange	Boolean	RW	BE	8.6.9.4.7
ConfigChangeTime	PTPtime	R	BE	8.6.9.4.9, 12.29.1.4
TickGranularity	Integer (tenths of nanoseconds)	R	BE	8.6.9.4.16
CurrentTime	PTPtime	R	BE	8.6.9.4.10, 12.29.1.4
ConfigPending	Boolean	R	BE	8.6.9.3, 8.6.9.4.8
ConfigChangeError	Integer	R	BE	8.6.9.3.1
SupportedListMax	Integer	R	BE	12.29.1.5

[a]R= Read only access; RW = Read/Write access.

[b]B = Required for Bridge or Bridge component support of enhancements for scheduled traffic.
E = Required for end station support of enhancements for scheduled traffic.

[c]The number of queues supported by a Port is available from the traffic class table (12.6.3)

12.29.1.1 The queueMaxSDUTable structure and data types

The queueMaxSDUTable consists of up to 8 entries, one per traffic class supported by the implementation, each entry consisting of a queueMaxSDU value (12.29.1.1.1) and a TransmissionOverrun counter.

12.29.1.1.1 queueMaxSDU

An unsigned integer value, denoting the maximum SDU size supported by the queue.

12.29.1.1.2 TransmissionOverrun

A counter that is incremented when the implementation detects that the transmission gate associated with a queue has closed and a frame that originated from the queue is still being transmitted by the MAC.

12.29.1.2 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 8-6. Each entry in the list is structured as a GateControlEntry (12.29.1.2.1).

12.29.1.2.1 GateControlEntry

A GateControlEntry consists of an operation name, followed by up to 2 parameters associated with the operation, as detailed in Table 8-6. The first parameter, if present, is a gateStatesValue (12.29.1.2.2); the second parameter, if present, is a timeIntervalValue (12.29.1.2.3).

12.29.1.2.2 gateStatesValue

A list of up to 8 tuples, one for each traffic class supported by the Port, each of which consists of a traffic class number in the range 0–7 and a gate state that is an enumerated value representing the gate state, *open* or *closed*, for that traffic class (see GateState in Table 8-6).

12.29.1.2.3 timeIntervalValue

An unsigned integer, denoting a TimeInterval in nanoseconds (see TimeInterval in Table 8-6).

12.29.1.3 RationalNumber

A rational number represented by an integer numerator and an integer denominator.

12.29.1.4 PTPtime

A time value, expressed as a PTP timescale (see 8.2 of IEEE Std 802.1AS-2011).

12.29.1.5 SupportedListMax

The maximum value supported by this Port of the AdminControlListLength and OperControlListLength parameters. It is available for use by schedule computation software to determine the port's control list capacity prior to computation.

17. Management Information Base (MIB)

17.2 Structure of the MIB

Insert a new subclause 17.2.22 at the end of 17.2, as shown, renumbering as necessary.

17.2.22 Structure of the IEEE8021-ST-MIB

The IEEE8021-ST-MIB provides for configuration of scheduled traffic (8.6.8, 8.6.8.4) on ports. Table 17-28 indicates the relationship between the SMIv2 objects defined in the MIB module (17.7.22) and managed objects defined in 12.29.

Table 17-28—IEEE8021-ST-MIB Structure and relationship to this standard

MIB table	MIB object	Reference
ieee8021STMaxSDU subtree		
ieee8021STMaxSDUTable		Max SDU table, 8.6.8.4, 8.6.9, 12.29.1
	ieee8021STTrafficClass	Traffic Class (Table index)
	ieee8021STMaxSDU	queueMaxSDU, 8.6.8.4, 8.6.9, 12.29.1, 12.29.1.1.1
	ieee8021TransmissionOverrun	TransmissionOverrun, 8.6.8.4, 8.6.9, 12.29.1, 12.29.1.1.2
ieee8021STParameters		
ieee8021STParametersTable		Scheduled Traffic parameter table, 8.6.8.4, 8.6.9, 12.29.1
	ieee8021STGateEnabled	GateEnabled, 8.6.8.4, 8.6.9, 8.6.9.4.14, 12.29.1
	ieee8021STAdminGateStates	AdminGateStates, 8.6.8.4, 8.6.9, 8.6.9.4.5, 12.29.1
	ieee8021STOperGateStates	OperGateStates, 8.6.8.4, 8.6.9, 8.6.9.4.22, 12.29.1
	ieee8021STAdminControlListLength	AdminControlListLength, 8.6.8.4, 8.6.9, 8.6.9.4.6, 12.29.1
	ieee8021STOperControlListLength	OperControlListLength, 8.6.8.4, 8.6.9, 8.6.9.4.23, 12.29.1
	ieee8021STAdminControlList	AdminControlList, 8.6.8.4, 8.6.9, 8.6.9.4.2, 12.29.1
	ieee8021STOperControlList	OperControlList, 8.6.8.4, 8.6.9, 8.6.9.4.19, 12.29.1
	ieee8021STAdminCycleTimeNumerator	Numerator — AdminCycleTime, 8.6.8.4, 8.6.9, 8.6.9.4.3, 12.29.1

Table 17-28—IEEE8021-ST-MIB Structure and relationship to this standard *(continued)*

MIB table	MIB object	Reference
	ieee8021STAdminCycleTimeDenominator	Denominator — AdminCycleTime, 8.6.8.4, 8.6.9, 8.6.9.4.3, 12.29.1
	ieee8021STOperCycleTimeNumerator	Numerator — OperCycleTime, 8.6.8.4, 8.6.9, 8.6.9.4.20, 12.29.1
	ieee8021STOperCycleTimeDenominator	Denominator — OperCycleTime, 8.6.8.4, 8.6.9, 8.6.9.4.20, 12.29.1
	ieee8021STAdminCycleTimeExtension	AdminCycleTimeExtension, 8.6.8.4, 8.6.9, 8.6.9.4.4, 12.29.1
	ieee8021STOperCycleTimeExtension	OperCycleTimeExtension, 8.6.8.4, 8.6.9, 8.6.9.4.21, 12.29.1
	ieee8021STAdminBaseTime	AdminBaseTime, 8.6.8.4, 8.6.9, 8.6.9.4.1, 12.29.1
	ieee8021STOperBaseTime	OperBaseTime, 8.6.8.4, 8.6.9, 8.6.9.4.18, 12.29.1
	ieee8021STConfigChange	ConfigChange, 8.6.8.4, 8.6.9, 8.6.9.4.7, 12.29.1
	ieee8021STConfigChangeTime	ConfigChangeTime, 8.6.8.4, 8.6.9, 8.6.9.4.9, 12.29.1
	ieee8021STTickGranularity	TickGranularity, 8.6.8.4, 8.6.9, 12.29.1
	ieee8021STCurrentTime	CurrentTime, 8.6.8.4, 8.6.9, 8.6.9.4.10, 12.29.1
	ieee802STConfigPending	ConfigPending, 8.6.9.4.8
	ieee8021STSupportedListMax	SupportedListMax, 12.29.1.5

17.3 Relationship to other MIBs

Insert a new subclause 17.3.23 at the end of 17.3, renumbering as necessary.

17.3.23 Relationship of the IEEE8021-ST-MIB to other MIBs

The IEEE8021-ST-MIB provides objects that extend the core management functionality of a Bridge, as defined by the IEEE8021-BRIDGE-MIB (17.7.2), in order to support the additional management functionality needed when the scheduled traffic extensions, as defined in 8.6.8.4, are supported by the Bridge. As support of the objects defined in the IEEE8021-ST-MIB also requires support of the IEEE8021-BRIDGE-MIB, the provisions of 17.3.2 apply to implementations claiming support of the IEEE8021-ST-MIB.

17.4 Security considerations

Insert a new subclause 17.4.23 at the end of 17.4, renumbering as necessary.

17.4.23 Security considerations of the IEEE8021-ST-MIB

There are a number of management objects defined in the IEEE8021-ST-MIB module that have a MAX-ACCESS clause of read-write. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a nonsecure environment without proper protection can have a negative effect on network operations.

The following tables and objects in the IEEE8021-ST-MIB can be misconfigured to interfere with the operation of the forwarding and queuing mechanisms in a manner that would be detrimental to the transmission of scheduled traffic:

ieee8021STMaxSDU
ieee8021STGateEnabled
ieee8021STAdminGateStates
ieee8021STAdminControlListLength
ieee8021STAdminControlList
ieee8021STAdminCycleTimeNumerator
ieee8021STAdminCycleTimeDenominator
ieee8021STAdminCycleTimeExtension
ieee8021STAdminBaseTime
ieee8021STConfigChange
ieee8021STConfigChangeTime

a) ieee8021STMaxSDUcan be misconfigured to affect the ability of a Port to transmit frames of greater than a given SDU size.
b) ieee8021STGateEnabled can be misconfigured to enable/disable scheduled traffic processing.
c) ieee8021STAdminGateStates can be misconfigured to affect the gate states of a Port on startup.
d) ieee8021STAdminControlListLength, ieee8021STAdminControlList, ieee8021STAdminCycleTimeNumerator, ieee8021STAdminCycleTimeDenominator, ieee8021STAdminCycleTimeExtension, ieee8021STAdminBaseTime, ieee8021STConfigChange, and ieee8021STConfigChangeTime can be misconfigured to affect the traffic schedule for the Port.

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not accessible) can be considered sensitive or vulnerable in some network environments. Thus it is important to control all types of access (including GET and/or NOTIFY) to these objects and possibly to encrypt the values of these objects when sending them over the network via SNMP.

17.7 MIB modules

Insert a new subclause 17.7.22 after 17.7.21, at the end of 17.7, as follows:

17.7.22 Definitions for the IEEE8021-ST-MIB module

```
IEEE8021-ST-MIB DEFINITIONS ::= BEGIN

-- ============================================================
-- MIB for support of the Scheduled Traffic Enhancements
-- for 802.1Q Bridges.
-- ============================================================


IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Unsigned32,
    Counter64
        FROM SNMPv2-SMI
    TEXTUAL-CONVENTION,
    TruthValue
        FROM SNMPv2-TC
    MODULE-COMPLIANCE,
    OBJECT-GROUP
        FROM SNMPv2-CONF
    ieee802dot1mibs
        FROM IEEE8021-TC-MIB
    ieee8021BridgeBaseComponentId,
    ieee8021BridgeBasePort
        FROM IEEE8021-BRIDGE-MIB
    ;

ieee8021STMib MODULE-IDENTITY
    LAST-UPDATED "201602190000Z" -- February 19, 2016
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
        "  WG-URL: www.ieee802.org/1
         WG-EMail: stds-802-1-L@ieee.org

           Contact: IEEE 802.1 Working Group Chair
            Postal: C/O IEEE 802.1 Working Group
                    IEEE Standards Association
                    445 Hoes Lane
                    Piscataway
                    NJ 08854
                    USA
             E-mail: stds-802-1-L@ieee.org"
    DESCRIPTION
        "The Bridge MIB module for managing devices that support
        the Scheduled Traffic Enhancements
        for 802.1Q Bridges.

        Unless otherwise indicated, the references in this MIB
        module are to IEEE Std 802.1Q-2014.

        Copyright (C) IEEE (2016).
        This version of this MIB module is part of IEEE802.1Q;
```

```
        see the draft itself for full legal notices."

    REVISION "201602190000Z" -- February 19, 2016
    DESCRIPTION
            "Initial version published as part of IEEE Std 802.1Qbv."

       ::= { ieee802dot1mibs 30 }

-- ================================================================
-- Textual Conventions
-- ================================================================

IEEE8021STTrafficClassValue ::= TEXTUAL-CONVENTION
    DISPLAY-HINT "d"
    STATUS      current
    DESCRIPTION
        "A traffic class value.
        This is the numerical value associated with a traffic
        class in a Bridge. Larger values are associated with
        higher priority traffic classes."
    REFERENCE   "12.29.1"
    SYNTAX      Unsigned32 (0..7)

IEEE8021STPTPtimeValue ::= TEXTUAL-CONVENTION
    STATUS      current
    DESCRIPTION
        "A PTPtime value, represented as a 48-bit unsigned integer
        number of seconds and a 32-bit unsigned integer number of
        nanoseconds.
        The first 6 octets represent the number of seconds: the
        first octet is the most significant
        octet of the 48-bit seconds value and the sixth octet
        is the least significant octet of the seconds value.
        The remaining octets, 7 through 10,  represent the
        number of nanoseconds: the seventh octet
        is the most significant octet of the 32-bit nanoseconds
        value and the tenth octet is the
        least significant octet of the nanoseconds value."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    SYNTAX      OCTET STRING (SIZE(10))


-- ================================================================
-- subtrees in the ST MIB
-- ================================================================

ieee8021STNotifications
    OBJECT IDENTIFIER ::= { ieee8021STMib 0 }

ieee8021STObjects
    OBJECT IDENTIFIER ::= { ieee8021STMib 1 }

ieee8021STConformance
    OBJECT IDENTIFIER ::= { ieee8021STMib 2 }

ieee8021STMaxSDUSubtree
    OBJECT IDENTIFIER ::= { ieee8021STObjects 1 }

ieee8021STParameters
```

```
    OBJECT IDENTIFIER ::= { ieee8021STObjects 2 }

-- ================================================================
-- The ieee8021STMaxSDUSubtree subtree
-- This subtree defines the objects necessary for the management
-- of the max SDU size parameters for each traffic class on a Port.
-- ================================================================

-- ================================================================
-- the ieee8021STMaxSDUTable
-- ================================================================

ieee8021STMaxSDUTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021STMaxSDUEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table containing a set of max SDU
        parameters, one for each traffic class.
        All writeable objects in this table must be
        persistent over power up restart/reboot."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STMaxSDUSubtree 1 }

ieee8021STMaxSDUEntry OBJECT-TYPE
    SYNTAX      Ieee8021STMaxSDUEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects containing Max SDU size
        for each traffic class supported by the Port."
    INDEX  { ieee8021BridgeBaseComponentId,
             ieee8021BridgeBasePort,
             ieee8021STTrafficClass  }
    ::= { ieee8021STMaxSDUTable 1 }

Ieee8021STMaxSDUEntry ::=
    SEQUENCE {
        ieee8021STTrafficClass
            IEEE8021STTrafficClassValue,
        ieee8021STMaxSDU
            Unsigned32,
        ieee8021TransmissionOverrun
            Counter64
        }

ieee8021STTrafficClass OBJECT-TYPE
    SYNTAX      IEEE8021STTrafficClassValue
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "The traffic class number associated with the row of
        the table.

        A row in this table is created for each traffic class
        that is supported by the Port"
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STMaxSDUEntry 1 }
```

```
ieee8021STMaxSDU OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "octets"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The value of the MaxSDU parameter for the traffic class.
        This value is represented as an unsigned integer. A value
        of 0 is interpreted as the max SDU size supported by
        the underlying MAC.

        The default value of the MaxSDU parameter is 0.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    DEFVAL { 0 }
    ::= { ieee8021STMaxSDUEntry 2}


ieee8021TransmissionOverrun OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A counter of transmission overrun events, where
        a PDU is still being transmitted by a MAC at the
        time when the transmission gate for the queue closed."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1, 12.29.1.1.2"
    DEFVAL { 0 }
    ::= { ieee8021STMaxSDUEntry 3}


-- ================================================================
-- The ieee8021STParameters subtree
-- This subtree defines the objects necessary for the management
-- of the traffic scheduling mechanism for IEEE Std 802.1Q.
-- ================================================================

-- ================================================================
-- the ieee8021STParametersTable
-- ================================================================

ieee8021STParametersTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021STParametersEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A table that contains the per-port manageable parameters for
        traffic scheduling.

        For a given Port, a row in the table exists.

        All writable objects in this table must be
        persistent over power up restart/reboot."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParameters 1 }
```

```
ieee8021STParametersEntry OBJECT-TYPE
    SYNTAX      Ieee8021STParametersEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
        "A list of objects that contains the manageable parameters for
        traffic scheduling for a port."
    INDEX  { ieee8021BridgeBaseComponentId,
             ieee8021BridgeBasePort
           }
    ::= { ieee8021STParametersTable 1 }

Ieee8021STParametersEntry ::=
    SEQUENCE {
        ieee8021STGateEnabled
            TruthValue,
        ieee8021STAdminGateStates
            OCTET STRING,
        ieee8021STOperGateStates
            OCTET STRING,
        ieee8021STAdminControlListLength
            Unsigned32,
        ieee8021STOperControlListLength
            Unsigned32,
        ieee8021STAdminControlList
            OCTET STRING,
        ieee8021STOperControlList
            OCTET STRING,
        ieee8021STAdminCycleTimeNumerator
            Unsigned32,
        ieee8021STAdminCycleTimeDenominator
            Unsigned32,
        ieee8021STOperCycleTimeNumerator
            Unsigned32,
        ieee8021STOperCycleTimeDenominator
            Unsigned32,
        ieee8021STAdminCycleTimeExtension
            Unsigned32,
        ieee8021STOperCycleTimeExtension
            Unsigned32,
        ieee8021STAdminBaseTime
            IEEE8021STPTPtimeValue,
        ieee8021STOperBaseTime
            IEEE8021STPTPtimeValue,
        ieee8021STConfigChange
            TruthValue,
        ieee8021STConfigChangeTime
            IEEE8021STPTPtimeValue,
        ieee8021STTickGranularity
            Unsigned32,
        ieee8021STCurrentTime
            IEEE8021STPTPtimeValue,
        ieee8021STConfigPending
            TruthValue,
        ieee8021STConfigChangeError
            Counter64,
        ieee8021STSupportedListMax
            Unsigned32
            }
```

```
ieee8021STGateEnabled OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The GateEnabled parameter determines whether traffic scheduling
        is active (true) or inactive (false).

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    DEFVAL { false }
    ::= { ieee8021STParametersEntry 1 }

ieee8021STAdminGateStates OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(1))
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative value of the GateStates parameter for the Port.
        The bits of the octet represent the gate states for the
        corresponding traffic classes; the MS bit corresponds to traffic class 7,
        the LS bit to traffic class 0. A bit value of 0 indicates closed; a
        bit value of 1 indicates open.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 2 }

ieee8021STOperGateStates OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(1))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The operational value of the GateStates parameter for the Port.
        The bits of the octet represent the gate states for the
        corresponding traffic classes; the MS bit corresponds to traffic class 7,
        the LS bit to traffic class 0. A bit value of 0 indicates closed; a
        bit value of 1 indicates open."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 3 }

ieee8021STAdminControlListLength OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative value of the ListMax parameter for the Port.
        The integer value indicates the number of entries (TLVs) in the
        AdminControlList.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 4 }

ieee8021STOperControlListLength OBJECT-TYPE
```

```
        SYNTAX      Unsigned32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
            "The operational value of the ListMax parameter for the Port.
            The integer value indicates the number of entries (TLVs) in the
            OperControlList."
        REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
        ::= { ieee8021STParametersEntry 5 }

    ieee8021STAdminControlList OBJECT-TYPE
        SYNTAX      OCTET STRING
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
            "The administrative value of the ControlList parameter for the Port.
            The octet string value represents the contents of the control list as
            an ordered list of entries, each encoded as a TLV, as follows.
            The first octet  of each TLV is interpreted as an
            unsigned integer representing a gate operation name:
                0: SetGateStates
                1-255: Reserved for future gate operations

            The second octet of the TLV is the length field,
            interpreted as an unsigned integer, indicating the number of
            octets of the value that follows the length. A length of
            zero indicates that there is no value
            (i.e., the gate operation has no parameters).

            The third through (3 + length -1)th octets encode the
            parameters of the gate operation, in the order that they
            appear in the definition of the operation
            in Table 8-6. Two parameter types are currently defined:

            - GateState:
                A GateState parameter is encoded in a single octet.
                The bits of the octet represent the gate states for the
                corresponding traffic classes; the MS bit corresponds
                to traffic class 7,
                the LS bit to traffic class 0. A bit value of 0 indicates
                closed; a bit value of 1 indicates open.

            - TimeInterval:
                A TimeInterval is encoded in 4 octets as a 32-bit
                unsigned integer, representing a number of nanoseconds.
                The first octet encodes the most significant 8 bits of the
                integer, and the fourth octet encodes the least
                significant 8 bits.

            The value of this object MUST be retained across
            reinitializations of the management system."
        REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
        ::= { ieee8021STParametersEntry 6 }

    ieee8021STOperControlList OBJECT-TYPE
        SYNTAX      OCTET STRING
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
```

```
        "The operational value of the ListMax parameter for the Port.
        The octet string value represents the contents of the control list as
        an ordered list of TLVs, as follows.
        The first octet  of each TLV is interpreted as a gate operation name:
            0: SetGateStates
            1-255: Reserved for future gate operations

        The second octet of the TLV is the length field,
        interpreted as an unsigned integer,
        indicating the number of octets of the value that follows
        the length. A length of zero indicates that there is no value
        (i.e., the gate operation has no parameters).

        The third through (3 + length -1)th octets encode the
        parameters of the gate operation, in the order that they
        appear in the definition of the operation
        in Table 8-6. Two parameter types are currently defined:

        - GateState:
            A GateState parameter is encoded in a single octet.
            The bits of the octet represent the gate states for the
            corresponding traffic classes; the MS bit corresponds to
            traffic class 7, the LS bit to traffic class 0.
            A bit value of 0 indicates closed; a
            bit value of 1 indicates open.

        - TimeInterval:
            A TimeInterval is encoded in 4 octets as a 32-bit
            unsigned integer, representing
            a number of nanoseconds. The first octet encodes the
            most significant 8 bits of the integer, and the fourth
            octet encodes the least significant 8 bits."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 7 }

ieee8021STAdminCycleTimeNumerator OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative value of the numerator of the CycleTime
        parameter for the Port.
        The numerator and denominator together represent the cycle time as
        a rational number of seconds.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 8 }

ieee8021STAdminCycleTimeDenominator OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative value of the denominator of the
        CycleTime parameter for the Port.
        The numerator and denominator together represent the cycle time as
        a rational number of seconds.
```

```
        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 9 }

ieee8021STOperCycleTimeNumerator OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The operational value of the numerator of the
        CycleTime parameter for the Port.
        The numerator and denominator together represent the cycle
        time as a rational number of seconds."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 10 }

ieee8021STOperCycleTimeDenominator OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The operational value of the denominator of the
        CycleTime parameter for the Port.
        The numerator and denominator together represent the
        cycle time as a rational number of seconds."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 11 }

ieee8021STAdminCycleTimeExtension OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "nanoseconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative value of the CycleTimeExtension
        parameter for the Port.
        The value is an unsigned integer number of nanoseconds.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 12 }

ieee8021STOperCycleTimeExtension OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "nanoseconds"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The operational value of the CycleTimeExtension parameter for the Port.
        The value is an unsigned integer number of nanoseconds."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 13 }

ieee8021STAdminBaseTime OBJECT-TYPE
    SYNTAX      IEEE8021STPTPtimeValue
    UNITS       "PTP time"
```

```
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The administrative value of the BaseTime parameter for the Port.
        The value is a representation of a PTPtime value,
        consisting of a 48-bit integer
        number of seconds and a 32-bit integer number of nanoseconds.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 14 }

ieee8021STOperBaseTime OBJECT-TYPE
    SYNTAX      IEEE8021STPTPtimeValue
    UNITS       "PTP time"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The operationsl value of the BaseTime parameter for the Port.
        The value is a representation of a PTPtime value,
        consisting of a 48-bit integer
        number of seconds and a 32-bit integer number of nanoseconds."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 15 }

ieee8021STConfigChange OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
        "The ConfigChange parameter signals the start of a
        configuration change
        when it is set to TRUE. This should only be done
        when the various administrative parameters
        are all set to appropriate values."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 16 }

ieee8021STConfigChangeTime OBJECT-TYPE
    SYNTAX      IEEE8021STPTPtimeValue
    UNITS       "PTP time"
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The PTPtime at which the next config change is scheduled to occur.
        The value is a representation of a PTPtime value,
        consisting of a 48-bit integer
        number of seconds and a 32-bit integer number of nanoseconds.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 17 }

    ieee8021STTickGranularity OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
```

```
    DESCRIPTION
        "The granularity of the cycle time clock, represented as an
        unsigned number of tenths of nanoseconds.

        The value of this object MUST be retained across
        reinitializations of the management system."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 18 }

 ieee8021STCurrentTime OBJECT-TYPE
    SYNTAX      IEEE8021STPTPtimeValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The current time, in PTPtime, as maintained by the local system.
        The value is a representation of a PTPtime value,
        consisting of a 48-bit integer
        number of seconds and a 32-bit integer number of nanoseconds."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 19 }

 ieee8021STConfigPending OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The value of the ConfigPending state machine variable.
        The value is TRUE if a configuration change is in progress
        but has not yet completed."
    REFERENCE   "8.6.8.4, 8.6.9.4, 12.29.1"
    ::= { ieee8021STParametersEntry 20 }

 ieee8021STConfigChangeError OBJECT-TYPE
    SYNTAX      Counter64
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "A counter of the number of times that a re-configuration
        of the traffic schedule has been requested with the old
        schedule still running and the requested base time was
        in the past."
    REFERENCE   "8.6.8.4, 8.6.9.3, 8.6.9.1.1, 12.29.1"
    ::= { ieee8021STParametersEntry 21 }

ieee8021STSupportedListMax OBJECT-TYPE
    SYNTAX      Unsigned32
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
        "The maximum value supported by this Port of the
        AdminControlListLength and OperControlListLength
        parameters."
    REFERENCE   "12.29.1.5"
    ::= { ieee8021STParametersEntry 22 }


-- ================================================================
-- IEEE8021 FQTSS MIB - Conformance Information
-- ================================================================
```

```
ieee8021STCompliances
    OBJECT IDENTIFIER ::= { ieee8021STConformance 1 }
ieee8021STGroups
    OBJECT IDENTIFIER ::= { ieee8021STConformance 2 }

-- ================================================================
-- units of conformance
-- ================================================================

-- ================================================================
-- the ieee8021STObjectsGroup group
-- ================================================================

ieee8021STObjectsGroup OBJECT-GROUP
    OBJECTS {
        ieee8021STMaxSDU,
        ieee8021TransmissionOverrun,
        ieee8021STGateEnabled,
        ieee8021STAdminGateStates,
        ieee8021STOperGateStates,
        ieee8021STAdminControlListLength,
        ieee8021STOperControlListLength,
        ieee8021STAdminControlList,
        ieee8021STOperControlList,
        ieee8021STAdminCycleTimeNumerator,
        ieee8021STAdminCycleTimeDenominator,
        ieee8021STOperCycleTimeNumerator,
        ieee8021STOperCycleTimeDenominator,
        ieee8021STAdminCycleTimeExtension,
        ieee8021STOperCycleTimeExtension,
        ieee8021STAdminBaseTime,
        ieee8021STOperBaseTime,
        ieee8021STConfigChange,
        ieee8021STConfigChangeTime,
        ieee8021STTickGranularity,
        ieee8021STCurrentTime,
        ieee8021STConfigPending,
        ieee8021STConfigChangeError,
        ieee8021STSupportedListMax
    }
    STATUS      current
    DESCRIPTION
        "Objects that allow management of scheduled traffic."
    ::= { ieee8021STGroups 1 }


-- ================================================================
-- compliance statements
-- ================================================================

ieee8021STCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
        "The compliance statement for devices supporting
        scheduled traffic.

        Support of the objects defined in this MIB module
        also requires support of the IEEE8021-BRIDGE-MIB; the
```

```
        provisions of 17.3.2 apply to implementations claiming
        support of this MIB. "

    MODULE -- this module
        MANDATORY-GROUPS {
            ieee8021STObjectsGroup
        }

    ::= { ieee8021STCompliances 1 }

END
```

Annex A

(normative)

PICS proforma—Bridge implementations[2]

A.5 Major capabilities

Insert the following row at the end of Table A.5:

Item	Feature	Status	References	Support
SCHED	Does the implementation support scheduled traffic?	O	5.4.1, 5.13.1, 8.6.8, 8.6.9, 12.29, 17.7.22	Yes [ ] No [ ]

A.14 Bridge management

Insert the following row at the end of Table A.14, renumbering item MGT-220 if necessary:

Item	Feature	Status	References	Support
MGT-248	Does the implementation support the management entities defined in 12.29?	SCHED: M	5.4.1 item ad), 12.29	Yes [ ] N/A [ ]

A.24 Management Information Base (MIB)

Insert the following row at the end of Table A.24, renumbering item MIB-41 if necessary:

Item	Feature	Status	References	Support
MIB-42	Is the IEEE8021-ST-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND SCHED: O	5.4.1 item ad), 12.29, 17.7.22	Yes [ ] N/A [ ] No [ ]

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

Insert a new Table A.44 at the end of Annex A as shown, renumbering as necessary:

A.44 Scheduled traffic

Item	Feature	Status	References	Support
	If scheduled traffic (SCHED in Table A.5) is not supported, mark N/A and ignore the remainder of this table.		5.4.1, 5.13.1, 8.6.8, 8.6.9, 12.29, 17.7.22	N/A[ ]
SCHED1	Support the state machines and associated definitions as specified in 8.6.9	SCHED:M	5.4.1, 5.13.1, 8.6.8, 8.6.9	Yes [ ] N/A [ ]
SCHED2	Does the implementation support the management entities defined in 12.29?	SCHED:M	5.4.1 item ad), 12.29	Yes [ ] N/A [ ]
SCHED3	Is the IEEE8021-ST-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND SCHED:O	5.4.1 item ad), 12.29, 17.7.22	Yes [ ] N/A [ ] No [ ]

Annex B

(normative)

PICS proforma—End station implementations[3]

B.5 Major capabilities

Insert the following row at the end of Table B.5:

Item	Feature	Status	References	Support	
SCHED	Does the implementation support scheduled traffic?	O	5.4.1, 5.13.1, 8.6.8, 8.6.9, 12.29, 17.7.22	Yes [ ]	No [ ]

Insert a new Table B.15 at the end of Annex B, renumbering as necessary:

B.15 Scheduled traffic

Item	Feature	Status	References	Support
	If scheduled traffic (SCHED in Table B.5) is not supported, mark N/A and ignore the remainder of this table.		5.4.1, 5.13.1, 8.6.8, 8.6.9, 12.29, 17.7.22	N/A[ ]
SCHED1	Support the state machines and associated definitions as specified in 8.6.9	SCHED:M	5.4.1, 5.13.1, 8.6.8, 8.6.9	Yes [ ] N/A [ ]
SCHED2	Does the implementation support the management entities defined in 12.29?	SCHED:M	5.4.1 item ad), 12.29	Yes [ ] N/A [ ]
SCHED3	Is the IEEE8021-ST-MIB module fully supported (per its MODULE-COMPLIANCE)?	MIB AND SCHED:O	5.4.1 item ad), 12.29, 17.7.22	Yes [ ] N/A [ ] No [ ]

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

Insert a new Annex Q as follows, renumbering as necessary.

Annex Q

(informative)

Traffic scheduling

This annex provides some background to the mechanisms provided in this standard for traffic scheduling, with the intent of providing some insight into the motivation for the provision of mechanisms for traffic scheduling and some of the ways that the mechanisms might be used.

Q.1 Motivation

Some applications have a need for frame delivery that is highly predictable in terms of the time at which frame transmission will occur, and the overall latency and jitter that will be experienced as the frame is propagated to its destination. Examples include industrial and automotive control applications, where data transmitted over the network is used to feed the parameters of control loops that are critical to the operation of the plant or machinery involved, and where frames carrying control data are transmitted on a repeating time schedule; late delivery of such frames can result in instability, inaccuracy, or failure of the operation of the control loops concerned. In some implementations, this need has been met by the provision of dedicated, highly engineered networks that are used solely for the transmission of time-critical control traffic; however, as the bandwidth occupied by such traffic is often low, and the cost of providing a dedicated control network can be high, it can be desirable to mix time-critical traffic with other classes of traffic in the same network, as long as such mixing can be achieved while still meeting the timing requirements of the time-critical traffic.

Prioritization alone is insufficient to address the needs of this kind of traffic; if a low priority frame is already being transmitted, then that transmission will complete before a higher priority frame can access the transmission medium, so there could be a delay of up to a maximum-sized frame before a high priority transmission can start. If such delays occur at every hop, then the accumulated latency could be unacceptably large.

Interfering traffic can itself be time-critical; for example, there could be multiple time-critical traffic streams related to different control systems. The ability to coordinate different time-critical traffic streams in such a way that they do not interfere with each other is also important.

One approach to meeting these objectives is to ensure that, at specific times, only one traffic class (or set of traffic classes) has access to the network; in effect to create a protected “channel” that is used by that traffic class alone. However, in order to ensure that the remaining unprotected traffic cannot affect the transmission of protected traffic, it is necessary to stop the transmission of unprotected traffic sufficiently far in advance of the protected time slot to be certain that the last unprotected transmission has completed before protected transmission starts. In the worst case, this would mean that the last unprotected transmission would start a maximum-sized frame transmission time before the start of the protected traffic “window.” In effect, a guard band is created before the time that the protected traffic transmission is due to start; transmission of unprotected traffic is not permitted between the start of the guard band and the start of the protected traffic window. The simplest approach is for the guard band to be as long as a maximum-sized frame transmission; however, the start of the guard band need not be fixed if the implementation is able to determine, from the size of the frames that it has queued, that there is sufficient time for a frame to be transmitted in its entirety before the start of the protected traffic window. This approach is illustrated in Figure Q-1. If the simple

approach were taken, then the time difference between T0 and T1 would be equal to the transmission time of a maximum-sized frame on that port (Figure Q-1, part A); if the implementation is able to determine frame transmission times on the fly, then T0 could vary between a maximum frame transmission time before T1 and a minimum frame transmission time before T1, depending upon what unprotected frames were available for transmission on the Port (Figure Q-1, part B).

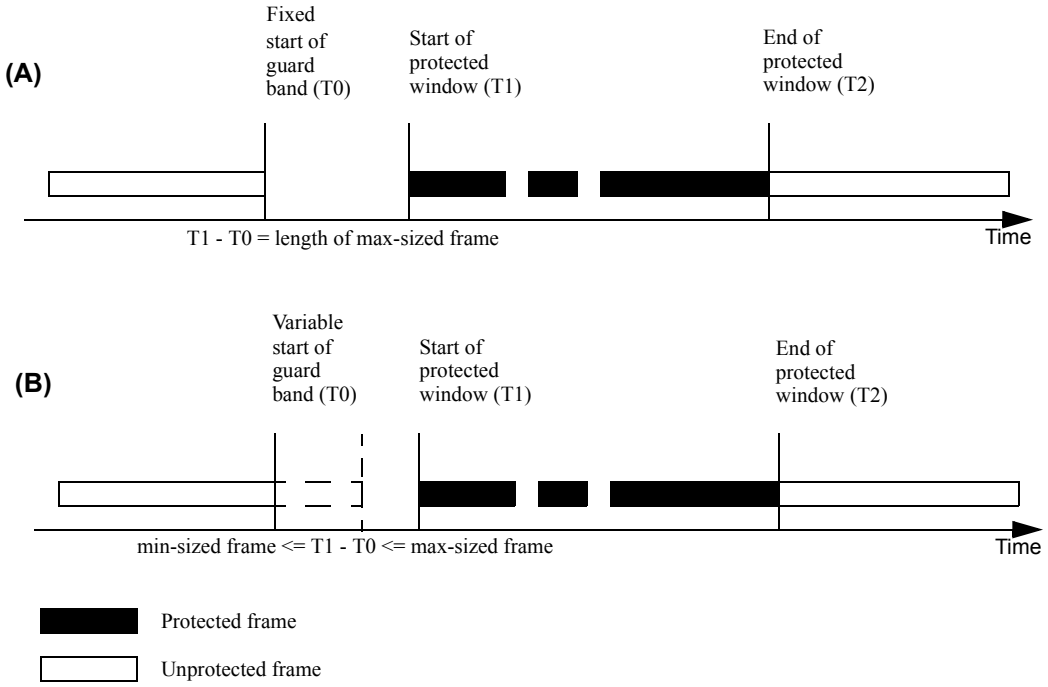


Figure Q-1—Establishing a guard band

Q.2 Using gate operations to create protected windows

The enhancements for scheduled traffic described in 8.6.8.4 allow transmission to be switched on and off on a timed basis for each traffic class that is implemented on a port. This switching is achieved by means of individual on/off transmission gates associated with each traffic class and a list of gate operations that control the gates; an individual SetGateStates operation has a time delay parameter that indicates the delay after the gate operation is executed until the next operation is to occur, and a GateState parameter that defines a vector of up to eight state values (open or closed) that is to be applied to each gate when the operation is executed. The gate operations allow any combination of open/closed states to be defined, and the mechanism makes no assumptions about which traffic classes are being "protected" and which are "unprotected"; any such assumptions are left to the designer of the sequence of gate operations. The sequence of gate operations terminates at the end of the list, and restarts when OperCycleTime (8.6.9.4.20) nanoseconds have elapsed since the start of the sequence, thus providing a repeating cycle of gate state changes. If OperCycleTime is shorter than the total execution time of the sequence of operations, the sequence is truncated and restarted at OperCycleTime.

There is no necessity to explicitly define the start of a guard band, as the specification of how the gates operate states that a frame for a given traffic class can only be transmitted if the gate associated with that traffic class is open and there is sufficient time for the whole of the frame to be transmitted before the gate is due to close. Figure Q-2 illustrates how gate operations could be used to create a protected window for traffic class 3 in order to achieve the result illustrated in Figure Q-1. A SetGateStates operation at time T1 sets the state of the gates for traffic classes 7, 6, 5, 4, 2, 1, and 0 to closed (C) and sets the state of the gate for

traffic class 3 to open (o), so only traffic class 3 can transmit after T1. At time T2, the SetGateStates operation reverses the state for all eight traffic classes, so traffic class 3 can no longer transmit after T1, but all other traffic classes can.

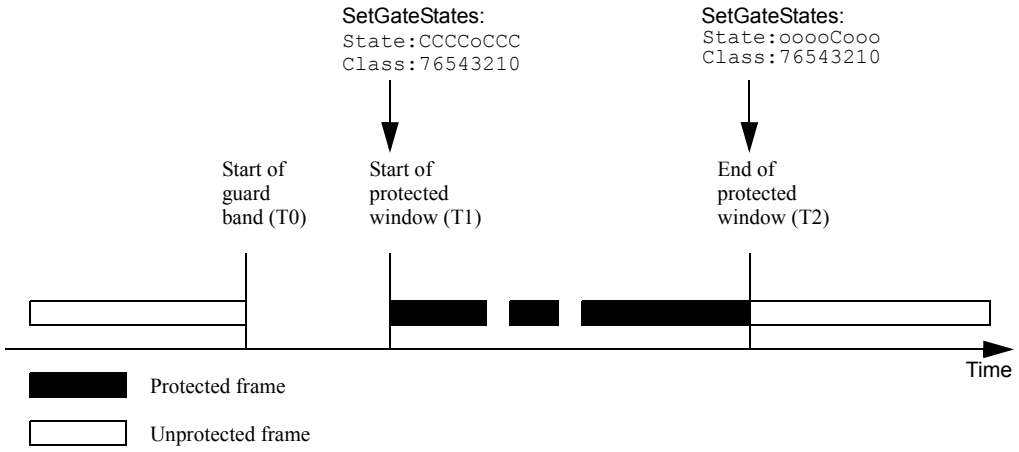


Figure Q-2—Using gate operations

There is no need to explicitly define the position of T0, because the "guard band" behavior is inherent in the defined behavior of the gates, as previously stated.

Q.3 Availability of PTP

The timing of gate operations as specified in 8.6.8.4 assumes that PTP is operating and therefore PTP time is available to the Bridge or end station. However, that is clearly not the case on initial startup or when PTP communication is interrupted for some reason; particular implementations could also choose to use alternative (non-PTP) mechanisms for ensuring that a common view of time is available across a number of systems. The approach adopted in these circumstances will be heavily dependent on the application, and so is an implementation choice; the mechanisms described in 8.6.8.4 are not themselves affected by that choice.

Q.4 Scheduled traffic and end stations

It is possible for Bridges to support the enhancements for scheduled traffic while end stations connected to those Bridges do not. In general, the scheduling mechanism is not needed in end stations that are only recipients of scheduled traffic.

Q.5 CycleTimeExtension variables

The AdminCycleTimeExtension (8.6.9.4.4) and OperCycleTimeExtension (8.6.9.4.21) state machine variables provide a means for an operator to control how a new gating cycle is configured into a system that is already running an existing gating cycle.

If the choice of cycle time for the new gating cycle (AdminCycleTime) is unchanged from the cycle time for the old gating cycle (OperCycleTime), and if the base time chosen for the new gating cycle (AdminBaseTime) is an integer multiple of the OperCycleTime different to OperBaseTime, then the new gating cycle will exactly "line up" with the old gating cycle, i.e., the cycle start times for the new gating cycle will be the same as they would have been for the old configuration. This could be considered to be the ideal case and allows the new gating cycle to be installed and executed with no timing issues.

However, if AdminCycleTime differs from OperCycleTime, and/or AdminBaseTime is in the future and is not an integer multiple of OperCycleTime different to OperBaseTime, then the old and new cycles do not line up, and when AdminBaseTime is reached (i.e., when the new configuration is installed and starts to execute), the last old cycle would normally be truncated in order to start the first new cycle. This could be undesirable if it results in a very short last old cycle; arguably it would be better to simply extend the penultimate old cycle by that small amount, rather than starting a very short cycle. The CycleTimeExtension variables allow this extension of the last old cycle to be done in a defined way; if the last complete old cycle would normally end less than OperCycleTimeExtension nanoseconds before the new base time, then the last complete cycle before AdminBaseTime is reached is extended so that it ends at AdminBaseTime.

Consensus

WE BUILD IT.

Connect with us on:

Facebook: https://www.facebook.com/ieeesa

Twitter: @ieeesa

LinkedIn: http://www.linkedin.com/groups/IEEESA-Official-IEEE-Standards-Association-1791118

IEEE-SA Standards Insight blog: http://standardsinsight.com

YouTube: IEEE-SA Channel

IEEE
standards.ieee.org
Phone: +1 732 981 0060 Fax: +1 732 562 1571