IEEE P802.1Qau/D2.4

Draft Standard for
Local and Metropolitan Area Networks—

Virtual Bridged Local Area Networks — Amendment: Congestion Notification

Sponsor

LAN MAN Standards Committee
of the
IEEE Computer Society

Prepared by the Data Center Bridging Task Group of IEEE 802.1

Abstract: This amendment specifies protocols, procedures and managed objects that support congestion management of long-lived data flows within network domains of limited bandwidth-delay product. This is achieved by enabling bridges to signal congestion to end stations capable of transmission rate limiting to avoid frame loss.
Keywords: local area networks, LANs, transparent bridging, MAC Bridges, VLANs, congestion, congestion notification.

DRAFT STATUS

Draft generated for first Sponsor ballot. It is identical to Draft 2.3, the Working Group draft approved by WG 802.1, except for the omission of the Editors' Forward, and Annex Z, and the numbers allocated for the MIB OUI (17.7.16), CNM PDU (33.3), and CNM TLV (33.5).

IEEE P802.1Qau/D2.4

Draft Standard for
Local and Metropolitan Area Networks —

Virtual Bridged Local Area Networks — Amendment: Congestion Notification

Sponsor

LAN MAN Standards Committee
of the
IEEE Computer Society

Prepared by the Data Center Bridging Task Group of IEEE 802.1

Abstract: This amendment specifies protocols, procedures and managed objects that support congestion management of long-lived data flows within network domains of limited bandwidth-delay product. This is achieved by enabling bridges to signal congestion to end stations capable of transmission rate limiting to avoid frame loss.
Keywords: local area networks, LANs, transparent bridging, MAC Bridges, VLANs, congestion, congestion notification.

Introduction to IEEE Std. 802.1Qau™

<< Editor's note: Per IEEE Standards Association editing policy, this introduction will be preceded, in the published version of IEEE Std. 802.1Qau, by a disclaimer to the effect that the Introduction is not a part of the Standard. >>

This Standard amends IEEE Std. 802.1Q-2005, providing congestion notification capabilities useful to Virtual Bridged Local Area Networks to support congestion management of long-lived data flows within network domains of limited bandwidth-delay product. Congestion notification mechanisms defined in this Standard include:

a) The ability of bridges and end stations to create Congestion Notification Domains for certain priority levels by signaling using the Link Layer Discovery Protocol defined in IEEE Std. 802.1AB-2005;
b) The ability for bridges to use priority remapping to automatically defend a Congestion Notification Domain against sources that are not aware of congestion notification;
c) Mechanisms by which bridges detect the congestion state of specified output queues, and send Congestion Notification Messages to the sources of a sampling of the frames in the queue;
d) Mechanisms by which an end station responds to Congestion Notification Messages by stopping, increasing, decreasing, or disabling control of the rate of output for frames; and
e) A set of managed objects to provide controls for these capabilities in both bridges and end stations.

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 can be obtained from

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Contents

List of Figures

List of Tables

IEEE P802.1Qau/D2.4

Draft Standard for
Local and Metropolitan Area Networks—
Amendment 7 to 802.1Q
Virtual Bridged Local Area Networks:
Congestion Notification

Editorial Note

This Standard amends IEEE Std 802.1Q to provide capabilities for congestion notification. Changes are applied to the base text generated by applying the amendments IEEE Std 802.1ad-2005, IEEE Std. 802.1ag-2007, IEEE Std. 802.1ak-2007, and IEEE Std. 802.1ap-2008 to IEEE Std 802.1Q-2005. Text shown in bold italics in this amendment defines the editing instructions necessary to changes to this base text. Three editing instructions are used: ***change***, ***delete***, and ***insert***. ***Change*** is used to make a change to existing material. The editing instruction specifies the location of the change and describes what is being changed. Changes to existing text may be clarified using ~~strikeout~~ markings to indicate removal of old material, and underscore markings to indicate addition of new material). ***Delete*** removes existing material. ***Insert*** adds new material without changing the existing material. Insertions may require renumbering. If so, renumbering instructions are given in the editing instruction. Editorial notes will not be carried over into future editions of IEEE Std. 802.1Q.

<< Editor's note: Before finalizing this Standard it will be necessary to produce a 'rolled up' base document comprising this Standard and 802.1Q-2005. This rolling up process usually reveals some unintended effects of what would otherwise appear a reasonable amendment. >>

1. Overview

Insert the following after the initial paragraphs of Clause 1.

This Standard specifies protocols, procedures and managed objects that support congestion management of long-lived data flows within network domains of limited bandwidth-delay product. This is achieved by enabling bridges to signal congestion to end stations capable of transmission rate limiting to avoid frame loss. This mechanism enables support for higher layer protocols that are highly loss or latency sensitive. VLAN tag encoded priority values are allocated to segregate frames subject to congestion control, allowing simultaneous support of both congestion controlled and other higher layer protocols. This Standard does not specify communication or reception of congestion notification information to or from end stations outside the congestion controlled domain or encapsulation of frames from those end stations across the domain.

1.1 Scope

Insert the following at end of subclause 1.1, relettering the bullet points so that they follow in order from those in the existing text.

This Standard specifies protocols, procedures, and managed objects to support congestion notification. These allow a Virtual Bridged Local Area Network or a portion thereof, with a limited bandwidth-delay product, to transfer long-lived data flows with a significantly reduced chance of frame loss compared to a network without congestion notification. To this end, it:

aa) Defines a means for VLAN-aware Bridges that support congestion notification to form Congestion Managed Domains within a Virtual Bridged LAN; and

ab) Defines a means for detecting congested queues in end stations and VLAN-aware Bridges, for signaling such congestion to the end stations sourcing the frames causing the congestion, and for those end stations to control the rate of transmission of those frames.

2. References

Insert the following references at the appropriate point:

IEEE Std 802.1AB™-2005, Station and Media Access Control Connectivity Discovery.

IETF RFC 2579 (STD 58), Textual Conventions for SMIv2.

3. Definitions

Insert the following definitions, renumbering to place them in the appropriate collating order.

3.1 Congestion Aware System: A bridge component or end station conforming to the congestion notification provisions of this standard.

3.2 Congestion Controlled Flow (CCF): A sequence of frames with the same priority, that the transmitting end station treats as belonging to a single flow, using a single Reaction Point to control the transmission rate of all those frames.

3.3 Congestion Notification Priority Value (CNPV): A value of the priority parameter that a congestion aware system uses to support congestion notification.

3.4 Congestion Notification Domain (CND): A connected subset of the end stations and VLAN-aware Bridges in a Virtual Bridged Network that are compatibly configured to support a Congestion Notification Priority Value. Multiple CNDs serving different Congestion Notification Priority Values can co-exist in a Virtual Bridged Network; the bridges, end stations and Ports included in the CNDs do not necessarily coincide.

3.5 Congestion Notification Message (CNM): A message transmitted by a Congestion Point to a Reaction Point, in response to a frame received from that Reaction Point, conveying congestion information used by the Reaction Point to reduce its transmission rate.

3.6 Congestion Notification Tag (CN-TAG): A tag that conveys a Flow Identifier, that a Reaction Point can add to transmitted Congestion Controlled Flow frames, and that a Congestion Point includes in a Congestion Notification Message.

3.7 Congestion Point (CP): A VLAN-aware Bridge or end station Port function (31.1.1) that monitors a single queue serving one or more Congestion Notification Priority Values, can generate Congestion Notification Messages, and can remove Congestion Notification Tags.

3.8 Flow Identifier (Flow ID): An identifier assigned by a congestion aware end station, unique within the scope of one or more of the source MAC addresses used by that system to transmit Congestion Controlled Flow frames, that can be used to associate each received Congestion Notification Message with the Reaction Point that rate controls the Congestion Controlled Flow that caused its transmission.

3.9 Reaction Point (RP): An end station port function (31.2.2.2) that controls the transmission rate of frames for one or more Congestion Controlled Flows, receiving and using Congestion Notification Messages as part of determining that rate.

4. Abbreviations

Insert the following definitions, placing them in the appropriate collating order (alphabetical).

CN	Congestion Notification
CCF	Congestion Controlled Flow
CN-TAG	Congestion Notification Tag
CND	Congestion Notification Domain
CNPV	Congestion Notification Priority Value
CNM	Congestion Notification Message
CP	Congestion Point
CPID	Congestion Point Identifier
Fb	Quantized Feedback
Flow ID	Flow Identifier
QCN	Quantized Congestion Notification protocol
RP	Reaction Point

5. Conformance

In subclause 5.4.1 VLAN-aware Bridge component options, insert the following additional bullet after current bullet (c):

d) Support congestion notification (5.4.3);

Insert the following subclause after Clause 5.4.2:

5.4.3 VLAN-aware Bridge requirements for congestion notification

A VLAN-aware Bridge implementation that conforms to the provisions of this standard for congestion notification (30, 31, 32, 33) shall:

a) Support, on one or more Ports, the creation of at least one Congestion Point (31.1.1);
b) Support, at each Congestion Point, the generation of Congestion Notification Messages (32.7) and the removal of Congestion Notification Tags (31.1.1);
c) Support the ability to configure the variables controlling the operation of each Congestion Point (12.17.1, 12.17.2, 12.17.3, 12.17.4);
d) Support the operation of each Port in each of the Congestion Notification Domain defense modes separately for each Congestion Notification Priority Value (32.1.1);
e) Conform to the required capabilities of the Link Layer Discovery Protocol (LLDP) of IEEE Std 802.1AB-2005 Clause 5.2; and
f) Support the use of the Congestion Notification TLV in LLDP (32.1.2).

A Provider Instance Port (PIP, 6.10) that conforms to the provisions of this standard for congestion notification shall:

g) Support Congestion Notification Message translation (32.16).

A Provider Edge Port (15.4) that conforms to the provisions of this standard for congestion notification shall:

h) Support Congestion Notification Message translation (32.16).

A VLAN-aware Bridge implementation that conforms to the provisions of this standard for congestion notification may:

i) Support the creation of up to seven Congestion Points on a Bridge Port (31.1.1); and/or
j) Support the IEEE8021-CN-MIB (17.7.16).

Insert the following subclause after Clause 5.9:

5.10 End station requirements for congestion notification

A VLAN-aware end station implementation that conforms to the provisions of this standard for congestion notification (30, 31, 32, 33) shall:

a) Support the creation of at least one Reaction Point (31.2.2.2);
b) Support, if Congestion Points are supported, the ability to configure the variables controlling the operation of each Congestion Point, if any (12.17.1, 12.17.2. 12.17.3. 12.17.4);
c) Support, at each supported Congestion Point, if any, the generation of Congestion Notification Messages (32.7);

d) Support the ability to configure the variables controlling the operation of each Reaction Point (12.17.1, 12.17.5, 12.17.6);
e) Support the operation of its Port in at least the cptDisabled and cptInterior Congestion Notification Domain defense modes separately for each Congestion Notification Priority Value (32.1.1);
f) Conform to the required specifications of the Link Layer Discovery Protocol (LLDP) of IEEE Std 802.1AB;
g) Support the use of the Congestion Notification TLV in LLDP (32.1.2);
h) Support, in each Reaction Point, the limiting of frames output in response to the reception of Congestion Notification Messages (31.2.2.2);
i) Support, if multiple Reaction Points are supported, the ability to distinguish among Congestion Notification Messages pertaining to different Reaction Points, and direct each such message to the correct Reaction Point (31.2.5);

A VLAN-aware end station implementation that conforms to the provisions of this standard for congestion notification (30, 31, 32, 33) may:

j) Support the creation of one or more Congestion Points (31.1.1);
k) Support the creation of more than one Reaction Point (31.2.2.2);
l) Support more than one Congestion Notification Priority Value per Reaction Point (31.2.1); and/or
m) Support the operation of its Port in both the cptInterior and cptInteriorReady Congestion Notification Domain defense modes separately for each Congestion Notification Priority Value (32.1.1);
n) Support the IEEE8021-CN-MIB (17.7.16).

6. Support of the MAC Service in VLANs

6.10.1 Data indications

Change the first paragraph as follows:

On receipt of an M_UNITDATA.indication primitive from the PIP-ISS, if the PIP is congestion aware (5.4.3) and the initial octets of the mac_service_data_unit contain a valid Congestion Notification Message encapsulation, the received frame is processed according to 32.16. Otherwise, the received frame shall be discarded if:

8. Principles of bridge operation

8.6.6 Queuing Frames

Insert the following paragraph at the end of subclause 8.6.6:

In a congestion aware Bridge (Clause 30), the act of queuing a frame for transmission on a Bridge Port can result in the Forwarding Process generating a Congestion Notification Message (CNM). The CNM is injected back into the Forwarding Process (Active topology enforcement, 8.6.1) as if it had been received on that Bridge Port.

12. Bridge management

12.1.1 Configuration management

Insert the following bullet item, relettered if necessary to follow the existing list.

g) The ability to create and delete the functional elements of congestion notification and to control their operation.

12.2 Managed objects

Insert the following bullet item, relettered if necessary to follow the existing list.

j) The congestion notification entities (12.17).

Insert a new Clause 12.17 as follows:

12.17 Congestion notification managed objects

A number of the variables that implement congestion notification, including CPs, RPs, and CND defense, are manageable objects[1]. There are a number of managed objects, each including a number of variables. The managed objects are the:

a) CN component managed object (12.17.1);
b) CN component priority managed object (12.17.2);
c) CN Port priority managed object (12.17.3);
d) Congestion Point managed object (12.17.4);
e) Reaction Point port priority managed object (12.17.5); and
f) Reaction Point group managed object (12.17.6).

NOTE—If multiple managed objects are altered over a period of time, then between the time the first and last object has been altered, operation of the state machines could produce unexpected results. This standard assumes that any number of managed objects can be altered as an atomic operation, so that no inconsistent intermediate states can occur. See 17.7.16 for one mechanism to ensure consistency.

12.17.1 CN component managed object

A single instance of the CN component managed object shall be implemented by a bridge component or end station that is congestion aware. It comprises all of the variables included in the CN component variables (32.2), as illustrated in Table 12-1. An end station may omit the managed objects noted "C" in the Conformance column of Table 12-1 if it does not support Congestion Points (CPs), and may in any case omit the managed object marked "e".

12.17.2 CN component priority managed object

The CN component priority managed object contains the managed objects that control a single CNPV for all Ports in an end station or bridge component. It comprises all of the variables included in the Congestion

1. The managed objects in this subclause are documented in a manner that is not parallel to that of the managed objects included in IEEE Std. 802.1Q-2005 and certain other of its amendments. It is intended that managed object definitions in future amendments to IEEE Std. 802.1Q will follow the format in this subclause. The resultant inconsistencies will be resolved in a future edition of IEEE Std. 802.1Q, insofar as this can be done without invalidating the MIB definitions in Clause 17, some of which depend on the format of Clause 12 used in IEEE Std. 802.1Q-2005.

Table 12-1—CN component managed object row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
cngMasterEnable	Boolean	RW	BE	32.2.1
cngCnmTransmitPriority	unsigned integer [0..7]	R	BC	32.2.2
cngDiscardedFrames	counter	R	BC	32.2.3
cngErroredPortList	list	R	Be	32.2.4

a. R = Read only access;
RW = Read/Write access
b. B = Required for a Bridge or Bridge component that is congestion aware;
C = Required for an end station that implements one or more CPs
E = Required for an end station that is congestion aware
e = Optional for an end station that is congestion aware

notification per-CNPV variables (32.3), as illustrated in Table 12-2. In one common use case for congestion notification, every Port of a bridge or end station has the same number of CPs, each configured for the same set of priority values. This managed object facilitates configuring that case. These objects can override, and can be overridden by, the CN Port priority managed objects (12.17.3) as described in 32.1.3.

Table 12-2—CN component priority managed object row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
cncpDefModeChoice	enum{cpcAdmin, cpcAuto}	RW	BE	32.3.1
cncpAlternatePriority	integer [0..7]	RW	BC	32.3.2
cncpAutoAltPri	integer [0..7]	R	BC	32.3.3
cncpAdminDefenseMode	enum{cptDisabled, cptInterior, cptInteriorReady, cptEdge}	RW	BE	32.3.4
cncpCreation	enum{cncpAutoEnable, cncpAutoDisable}	RW	BE	32.3.5
cncpLldpInstanceChoice	enum{cnlNone, cnlAdmin}	RW	BE	32.3.6
cncpLldpInstanceSelector	802.1AB LLDP instance selector	RW	BE	32.3.7

a. R = Read only access;
RW = Read/Write access
b. B = Required for a Bridge or Bridge component that is congestion aware;
C = Required for an end station that implements one or more CPs
E = Required for an end station that is congestion aware

A CN component priority managed object shall be implemented by a bridge component or end station that is congestion aware for each priority value that can be a CNPV. An end station may omit the managed objects noted “C” in the Conformance column in Table 12-2 if it does not support Congestion Points (CPs). The operations that can be performed on a congestion aware system’s CN component priority managed object are:

a) Create CN component priority managed object (12.17.2.1); and
b) Delete CN component priority managed object (12.17.2.2).

12.17.2.1 Create CN component priority managed object

Creating a CN component priority managed object creates an instance of each of the Congestion notification per-CNPV variables (32.3), and also creates the corresponding CN Port priority managed objects (12.17.3) and all their dependent managed objects and variables, on every Port in the bridge component or end station, as illustrated in CN Port priority managed object. Depending on the value of cncpCreation (32.3.5), creating a CN component priority managed object can make the selected priority a CNPV throughout the bridge component or end station.

12.17.2.2 Delete CN component priority managed object

Deleting a CN component priority managed object deletes all of the CN component variables (32.2), and also deletes the corresponding CN Port priority managed objects (12.17.3) and all their dependent managed objects and variables, on all Ports in the bridge component or end station, thus making the priority not a CNPV throughout the bridge component or end station.

12.17.3 CN Port priority managed object

There is one CN Port priority managed object per Port per priority in a congestion aware end station or bridge component. It comprises some of the variables included in the CND defense per-Port per-CNPV variables (32.4), as illustrated in Table 12-3. These objects can override, and can be overridden by, the CN component managed objects (12.17.1) and CN component priority managed objects (12.17.2) as described in 32.1.3.

Table 12-3—CN Port priority managed object row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
cnpdDefModeChoice	enum{cpcAdmin, cpcAuto, cpcComp}	RW	BE	32.4.1
cnpdAdminDefenseMode	enum{cptDisabled, cptInterior, cptInteriorReady, cptEdge}	RW	BE	32.4.2
cnpdAutoDefenseMode	enum{cptDisabled, cptInterior, cptInteriorReady, cptEdge}	R	BE	32.4.3
cnpdLldpInstanceChoice	enum{cnlNone, cnlAdmin}	RW	BE	32.4.4
cnpdLldpInstanceSelector	802.1AB LLDP instance selector	RW	BE	32.4.5
cnpdAlternatePriority	integer [0..7]	RW	BC	32.4.6

a. R = Read only access;
RW = Read/Write access
b. B = Required for a Bridge or Bridge component that is congestion aware;
C = Required for an end station that implements one or more CPs
E = Required for an end station that is congestion aware

A CN Port priority managed object is created or deleted when the corresponding Port or CN component priority managed object is created or deleted, and its initial state on creation is determined by cncpCreation (32.3.5).

The CN Port priority managed object shall be implemented by a bridge component or end station that is congestion aware. An end station may omit the managed objects noted "C" in the Conformance column in Table 12-3 if it does not support Congestion Points (CPs).

12.17.4 Congestion Point managed object

There is one Congestion Point managed object for each CP in a bridge component or end station. It comprises some of the variables included in the Congestion Point variables (32.8), as illustrated in Table 12-4.

Table 12-4—Congestion Point managed object row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
cpMacAddress	MAC address	R	BC	32.8.1
cpId	octet string (size = 8)	R	BC	32.8.2
cpQSp	unsigned integer	R	BC	32.8.3
cpW	real number	RW	BC	32.8.6
cpSampleBase	unsigned integer	RW	BC	32.8.11
cpDiscardedFrames	counter	R	BC	32.8.12
cpTransmittedFrames	counter	R	BC	32.8.13
cpTransmittedCnms	counter	R	BC	32.8.14
cpMinHeaderOctets	unsigned integer	RW	BC	32.8.15

a. R = Read only access;
RW = Read/Write access
b. B = Required for a Bridge or Bridge component that is congestion aware;
C = Required for an end station that implements one or more CPs

The Congestion Point managed object shall be implemented by a congestion aware bridge component. It shall be implemented by an end station that supports Congestion Points (CPs).

NOTE—The Recommended priority to traffic class mappings in Table 8-2 can assign more than one CNPV to the same traffic class, the same queue, and hence the same CP. There can be only one CP controlling a given queue, and that CP has one set of controlling managed objects, not one set per CNPV. That set of managed objects can be accessed using any of the CNPV values assigned to the CP's queue. Thus, changing a managed object for one CNPV changes the managed object for all CNPVs assigned to the same queue.

12.17.5 Reaction Point port priority managed object

A congestion aware end station shall implement one Reaction Point port priority managed object for each CNPV on each port. The Reaction Point port priority managed object controls the creation of RPs on the port and CNPV, as illustrated in Table 12-5.

Table 12-5—Reaction Point port priority managed object row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
rpppMaxRps	unsigned integer	RW	E	32.10.1
rpppCreatedRps	counter	R	E	32.10.2
rpppRpCentiseconds	unsigned integer	R	E	32.10.3

a. R = Read only access;
RW = Read/Write access
b. E = Required for an end station that is congestion aware

12.17.6 Reaction Point group managed object

There is one Reaction Point group managed object for each set of Reaction Point group variables (32.11), as illustrated in Table 12-6. The Reaction Point group managed object shall be implemented by an end station that is congestion aware.

Table 12-6—Reaction Point group managed object row elements

Name	Data type	Operations supported[a]	Conformance[b]	References
rpgEnable	Boolean	RW	E	32.11.1
rpgTimeReset	unsigned integer	RW	E	32.11.2
rpgByteReset	unsigned integer	RW	E	32.11.3
rpgThreshold	unsigned integer	RW	E	32.11.4
rpgMaxRate	unsigned integer	RW	E	32.11.5
rpgAiRate	unsigned integer	RW	E	32.11.6
rpgHaiRate	unsigned integer	RW	E	32.11.7
rpgGd	real number	RW	E	32.11.8
rpgMinDecFac	real number	RW	E	32.11.9
rpgMinRate	unsigned integer	RW	E	32.11.10

a. RW = Read/Write access
b. E = Required for an end station that is congestion aware

17. MIB Modules

17.1 The Internet Standard Management Framework

17.2 Structure of the MIB

Insert the following line at the appropriate place in 17-1:

Table 17-1—Structure of the MIB Modules

Module	subclause	Defining standard	Reference	Notes
IEEE8021-CN-MIB	17.7.16	802.1Qau	30	Initial version in 802.1Qau

Insert the following subclause after Clause 17.2.15:

17.2.16 Structure of the Congestion Notification MIB

Clause 12.17 of this document defines the information model associated with this standard in a protocol independent manner. Table 17-2 describes the relationship between the SMIv2 objects defined in the MIB module in 17.7.16 and the variables and managed objects defined in Clause 12 and Clause 32.

Table 17-2—Variables, managed object tables, and MIB objects

Clause 32 table or variable	Clause 12/ Clause 32 reference	MIB object (17.7)
CN component managed object, CN component variables	**12.17.1 32.2**	**ieee8021CnGlobalTable**
cngMasterEnable	32.2.1	ieee8021CnGlobalMasterEnable
cngCnmTransmitPriority	32.2.2	ieee8021CnGlobalCnmTransmitPriority
cngDiscardedFrames	32.2.3	ieee8021CnGlobalDiscardedFrames
cngErroredPortList	32.2.4	ieee8021CnErroredPortTable
CN component priority managed object, Congestion notification per-CNPV variables	**12.17.2 32.3**	**ieee8021CnCompntPriTable**
cncpDefModeChoice	32.3.1	ieee8021CnComPriDefModeChoice
cncpAlternatePriority	32.3.2	ieee8021CnComPriAlternatePriority
cncpAutoAltPri	32.3.3	ieee8021CnComPriAutoAltPri
cncpAdminDefenseMode	32.3.4	ieee8021CnComPriAdminDefenseMode
cncpCreation	32.3.5	ieee8021CnComPriCreation
cncpLldpInstanceChoice	32.3.6	ieee8021CnComPriLldpInstanceChoice
cncpLldpInstanceSelector	32.3.7	ieee8021CnComPriLldpInstanceSelector

Table 17-2—Variables, managed object tables, and MIB objects

Clause 32 table or variable	Clause 12/ Clause 32 reference	MIB object (17.7)
CN Port priority managed object, CND defense per-Port per-CNPV variables	**12.17.2 32.4**	**ieee8021CnPortPriTable**
cnpdDefModeChoice	32.4.1	ieee8021CnPortPriDefModeChoice
cnpdAdminDefenseMode	32.4.2	ieee8021CnPortPriAdminDefenseMode
cnpdAutoDefenseMode	32.4.3	ieee8021CnPortPriAutoDefenseMode
cnpdLldpInstanceChoice	32.4.4	ieee8021CnPortPriLldpInstanceChoice
cnpdLldpInstanceSelector	32.4.5	ieee8021CnPortPriLldpInstanceSelector
cnpdAlternatePriority	32.4.6	ieee8021CnPortPriAlternatePriority
Congestion Point managed object, Congestion Point variables	**12.17.4 32.8**	**ieee8021CnCpTable**
cpMacAddress	32.8.1	ieee8021CnCpMacAddress
cpId	32.8.2	ieee8021CnCpIdentifier
cpQSp	32.8.3	ieee8021CnCpQueueSizeSetPoint
cpW	32.8.6	ieee8021CnCpFeedbackWeight
cpSampleBase	32.8.11	ieee8021CnCpMinSampleBase
cpDiscardedFrames	32.8.12	ieee8021CnCpDiscardedFrames
cpTransmittedFrames	32.8.13	ieee8021CnCpTransmittedFrames
cpTransmittedCnms	32.8.14	ieee8021CnCpTransmittedCnms
cpMinHeaderOctets	32.8.15	ieee8021CnCpMinHeaderOctets
(None)[a]	**(none)**	**ieee8021CnCpidToInterfaceTable**
Reaction Point port priority managed object, Reaction Point per-Port per-CNPV variables	**12.17.5 32.10**	**ieee8021CnRpPortPriTable**
rpppMaxRps	32.10.1	ieee8021CnRpPortPriMaxRps
rpppCreatedRps	32.10.2	ieee8021CnRpPortPriCreatedRps
rpppRpCentiseconds	32.10.3	ieee8021CnRpPortPriCentiseconds

Table 17-2—Variables, managed object tables, and MIB objects

Clause 32 table or variable	Clause 12/ Clause 32 reference	MIB object (17.7)
Reaction Point group managed object, Reaction Point group variables	**12.17.6 32.11**	**ieee8021CnRpGroupTable**
rpgEnable	32.11.1	ieee8021CnRpgEnable
rpgTimeReset	32.11.2	ieee8021CnRpgTimeReset
rpgByteReset	32.11.3	ieee8021CnRpgByteReset
rpgThreshold	32.11.4	ieee8021CnRpgThreshold
rpgMaxRate	32.11.5	ieee8021CnRpgMaxRate
rpgAiRate	32.11.6	ieee8021CnRpgAiRate
rpgHaiRate	32.11.7	ieee8021CnRpgHaiRate
rpgGd	32.11.8	ieee8021CnRpgGd
rpgMinDecFac	32.11.9	ieee8021CnRpgMinDecFac
rpgMinRate	32.11.10	ieee8021CnRpgMinRate

a. This table is an artifact of SNMP, required to find an entry in the ieee8021CnCpTable, given a CPID (32.8.2, 33.4.4).

17.3 Relationship to other MIB modules

Insert the following subclause after Clause 17.3.15:

17.3.16 Relationship of the Congestion Notification MIB to other MIB modules

17.3.16.1 Interface MIB

Clause 17.7.16 defines a Congestion Notification MIB (CN MIB) module that supports congestion notification with the textual conventions imported from the TC MIB in 17.7.17. A system implementing the CN MIB module in Clause 17.7.16 shall also implement at least the System Group of the SNMPv2-MIB defined in IETF RFC 3418 and the Interfaces Group (the Interfaces MIB module, or IF-MIB) defined in IETF RFC 2863. The Interfaces Group has one conceptual row in a table for every interface in a system. Section 3.3 of IETF RFC 2863, the Interface MIB Evolution, defines hierarchical relationships among interfaces. IETF RFC 2863 also requires that any MIB module that is an adjunct of the Interface Group clarify specific areas within the Interface MIB module. These areas were intentionally left vague in IETF RFC 2863 to avoid over constraining the MIB, thereby precluding management of certain media types. These areas are clarified in other clauses which define the MIB modules in this standard. Even if a system supports none of these, if it supports the CN MIB module, and hence, the Interfaces Group, the clarifications from the other clauses shall be applied to the Interfaces Group. The relationship between IETF RFC 2863 and IETF RFC 3418 interfaces and ports is also described in previous subclauses of 17.3. In addition, if both the CN MIB module (17.7.16) and IEEE Std 802.3, Clause 43, are supported, a bridge component or end station may:

a) Assign each IEEE 802.3 port in the aggregation its own conceptual row in the IETF RFC 2863 IF-MIB; and

b) Assign one conceptual row in the IETF RFC 2863 IF-MIB that references the aggregated port. This can be the same as the interface identifying the port. It shall not be the same as any of the IEEE 802.3 ports being aggregated.

17.3.16.2 IEEE8021-CFM and IEEE8021-CFM-V2 MIBs

Clause 31.1 allows a system that supports either the dot1agCfmVlanTable from the IEEE8021-CFM MIB module or the ieee8021CfmVlanTable from the IEEE8021-CFM-V2 MIB module to use those tables to select the vlan_identifier for a transmitted Congestion Notification Message.

17.4 Security considerations

Insert the following subclause after Clause 17.4.15:

17.4.16 Security considerations of the Congestion Notification MIB

There are a number of management objects defined in the IEEE8021-CN-MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These tables and objects and their sensitivity/ vulnerability are described below.

Improper manipulation of certain tables and objects can result in the misidentification of the boundaries of Congestion Notification Domains (CNDs, 30.6). Any such errors can result in the following problems:

a) Frames not originated through a Reaction Point (RP, 31.2.2.2) can pass through Congestion Points (CPs, 31.1.1). The inability to throttle these frames via Congestion Notification Messages (CNM, 33.3) can cause legitimate, Reaction Point-originated, frames to experience unacceptably high loss rates.
b) Frames not originating through a Reaction Point (RP, 31.2.2.2) can pass through queues in the Bridged Network that are not Congestion Points (CPs, 31.1.1), where they can encounter uncontrolled congestion. This can cause those frames to experience unacceptably high loss rates.
c) The Congestion Notification Tag (CN-TAG, 30.5) in a data frame or CNM can be improperly discarded. This can cause that CNM, or a CNM resulting from the data frame, to be discarded by the end station that sourced the frame. As a result, that flow, and all flows passing through the same CPs, can experience unacceptably high loss rates.
d) The CN-TAG in a data frame or CNM can be improperly retained. This can result in a failure of two end stations to communicate, because the receiving end station is unable to process frames with CN-TAGs.

The following tables and objects in the IEEE8021-CN-MIB can cause such CND boundary identification errors:

ieee8021CnGlobalMasterEnable
ieee8021CnComPriDefModeChoice
ieee8021CnComPriAdminDefenseMode
ieee8021CnComPriCreation
ieee8021CnComPriLldpInstanceChoice
ieee8021CnComPriLldpInstanceSelector
ieee8021CnComPriRowStatus
ieee8021CnPortPriDefModeChoice
ieee8021CnPortPriAdminDefenseMode
ieee8021CnPortPriLldpInstanceChoice

ieee8021CnPortPriLldpInstanceSelector

Improper manipulation of certain other objects can result in assigning a data frame or CNM to the incorrect priority. This can result in the affected data streams experiencing either loss rates that are either higher or lower than expected by the network administrator. Lower than expected loss rates for one data stream can cause higher than expected loss rates for other streams. These affects are typically, but not universally, less severe than the loss rate anomalies caused by CND boundary misidentification. The following variables can cause improper priority assignment:

ieee8021CnGlobalCnmTransmitPriority
ieee8021CnComPriAlternatePriority
ieee8021CnPortPriAlternatePriority

Improper manipulation of the remainder of the read-write and read-create objects can cause the congestion notification algorithm to operate in ways not intended by the network administrator. This can result in excessively high data frame loss rates and/or low link utilization rates. These variables are the following:

ieee8021CnCpQueueSizeSetPoint
ieee8021CnCpFeedbackWeight
ieee8021CnCpMinSampleBase
ieee8021CnCpMinHeaderOctets
ieee8021CnRpPortPriMaxRps
ieee8021CnRpgEnable
ieee8021CnRpgTimeReset
ieee8021CnRpgByteReset
ieee8021CnRpgThreshold
ieee8021CnRpgMaxRate
ieee8021CnRpgAiRate
ieee8021CnRpgHaiRate
ieee8021CnRpgGd
ieee8021CnRpgMinDecFac
ieee8021CnRpgMinRate

Unintended access to any of the readable tables or variables in the IEEE8021-CN-MIB alerts the reader that congestion notification is configured, and (for all tables and variables except those in the ieee8021CnGlobalTable and ieee8021CnCpidToInterfaceTable) on which priority value or values congestion notification is configured. This information can suggest to an attacker what applications are being run, and thus suggest application-specific attacks, or to enable the attacker to detect whether their attacks are being successful or not. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP.

17.7 MIB modules

Insert the following subclause after Clause 17.5.15:

17.7.16 Congestion Notification MIB module

In the MIB definition below, if any discrepancy between the DESCRIPTION text and the corresponding definition in Clause 12 occur, the definition in Clause 12 takes precedence.

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

-- ********************************************************************
-- IEEE P802.1Qau(TM) Congestion Management MIB
-- ********************************************************************

IMPORTS
    MODULE-IDENTITY,
    OBJECT-TYPE,
    Integer32,
    Counter32,
    Counter64,
    Unsigned32              FROM SNMPv2-SMI    -- [RFC2578]
    TEXTUAL-CONVENTION,
    TimeInterval,
    RowStatus,
    TruthValue,
    MacAddress              FROM SNMPv2-TC     -- [RFC2579]
    MODULE-COMPLIANCE,
    OBJECT-GROUP            FROM SNMPv2-CONF   -- [RFC2580]
    InterfaceIndex,
    ifGeneralInformationGroup
                            FROM IF-MIB        -- [RFC2863]
    IEEE8021PriorityValue,
    IEEE8021PbbComponentIdentifier
                            FROM IEEE8021-TC-MIB -- [IEEE 802.1ap]
    LldpV2DestAddressTableIndex
                            FROM LLDP-V2-TC-MIB
    systemGroup             FROM SNMPv2-MIB    -- [RFC3418]
   ;

ieee8021CnMib MODULE-IDENTITY
    LAST-UPDATED "200910280000Z"    -- 10/28/2009 00:00GMT
    ORGANIZATION "IEEE 802.1 Working Group"
    CONTACT-INFO
       "WG-URL:   http://grouper.ieee.org/groups/802/1/index.html
        WG-EMail: stds-802-1@ieee.org

        Contact:  Norman Finn

                  Cisco Systems
                  170 W. Tasman Drive
                  San Jose, CA 95134, USA

        E-mail:   nfinn@cisco.com
       "
   DESCRIPTION
     "Congestion notification module for managing IEEE 802.1Qau"
    REVISION       "200910280000Z"    -- 10/28/2009 00:00GMT
    DESCRIPTION
      "Included in IEEE P802.1Qau Draft 2.4

        Copyright (C) IEEE."
    ::= { iso(1) org(3) ieee(111)
```

```
            standards-association-numbers-series-standards (2)
            lan-man-stds (802) ieee802dot1 (1) ieee802dot1mibs (1) 18 }

    ieee8021CnMIBObjects      OBJECT IDENTIFIER ::= { ieee8021CnMib 1 }
    ieee8021CnConformance     OBJECT IDENTIFIER ::= { ieee8021CnMib 2 }

    -- ****************************************************************
    -- Textual conventions
    -- ****************************************************************

    Ieee8021CnControlChoice ::= TEXTUAL-CONVENTION
        STATUS  current
        DESCRIPTION
           "This controls what other object selects the CND defense mode and
            the Congestion Notification Priority Value (CNPV) alternate
            priority for a CNPV in an end station or bridge component, or
            for a CNPV on a particular Port in an end station or bridge
            component.  It can take the following values:

            cpcAdmin(1) The CND defense mode and alternate priority are
                        controlled by the administrative variables in the
                        same table entry as this object.
            cpcAuto(2)  This Port or all Ports' CND defense modes are
                        controlled automatically, as indicated by
                        ieee8021CnPortPriAutoDefenseMode, and the
                        alternate priority by ieee8021CnComPriAutoAltPri.
            cpcComp(3)  This CND defense mode and alternate priority are
                        both controlled by ieee8021CnPortPriTable.
           "
        REFERENCE
           "IEEE 802.1Qau clauses 32.3.1, 32.4.1, Table 32-2"
        SYNTAX  INTEGER {
            cpcAdmin    (1),
            cpcAuto     (2),
            cpcComp     (3)
        }

    Ieee8021CnDefenseMode ::= TEXTUAL-CONVENTION
        STATUS  current
        DESCRIPTION
           "For a given Congestion Notification Priority Value (CNPV), a
            port can operate in one of four CND defense modes.  The CND
            defense mode determines whether congestion notification is
            enabled or not, and if enabled, whether the port translates the
            CNPV to a non-CNPV value on input, and whether the port removes
            CN-TAGs on output.

            cptDisabled(1)      The congestion notification capability is
                                administratively disabled for this priority
                                value and port. This priority is not a CNPV.
                                The priority regeneration table controls the
                                remapping of input frames on this port for
                                this priority. CN-TAGs are neither added by
                                an end station nor stripped by a bridge.
```

```
            cptInterior(2)      On this port and for this CNPV, the priority
                                parameters of input frames are not remapped,
                                regardless of the priority regeneration
                                table. CN-TAGs are not added by an end
                                station, and are removed from frames before
                                being output by a bridge.
            cptInteriorReady(3) On this port and for this CNPV, the priority
                                parameters of input frames are not remapped,
                                regardless of the priority regeneration
                                table. CN-TAGs can be added by an end
                                station, and are not removed from frames by
                                a bridge.
            cptEdge(4)          On this port and for this CNPV, the priority
                                parameters of input frames are remapped to
                                an alternate (non-CNPV) value, regardless of
                                the priority regeneration table. CN-TAGs are
                                not added by an end station, and are removed
                                from frames before being output by a bridge.
                                This mode is optional for an end station.
          "
        REFERENCE
          "IEEE 802.1Qau clause 32.1.1, 32.3.4, 32.4.2, 32.4.3, Table 32-2"
        SYNTAX  INTEGER {
            cptDisabled      (1),
            cptInterior      (2),
            cptInteriorReady (3),
            cptEdge          (4)
        }

    Ieee8021CnLldpChoice ::= TEXTUAL-CONVENTION
        STATUS current
        DESCRIPTION
          "Specifies how to determine what index value is to be used as
           the index to lldpDestAddressTable, the table of destination MAC
           addresses, for both the destination addresses on transmitted
           LLDPDUs and on received LLDPDUs, found in the LLDP-MIB, either:

               cnlNone(1)      No LLDP Congestion Notification TLV is to
                               carry Per-priority CNPV indicators or
                               Per-priority Ready indicators on this Port
                               for this priority (or all Ports and all
                               priorities, as appropriate to the managed
                               object).
               cnlAdmin(2)     The administrative LLDP instance selector
                               in the same table entry as this object
                               governs which LLDP instance will carry the
                               Per-priority CNPV indicators and
                               Per-priority Ready indicators for this
                               priority in its Congestion Notification TLV
                               on this Port (or all Ports and all
                               priorities, as appropriate to the managed
                               object).
               cnlComponent(3) ieee8021CnComPriLldpInstanceSelector governs
                               LLDP instance selection for this Port and
```

```
                                priority.
            "
        REFERENCE
            "IEEE 802.1Qau clause 32.3.6, 32.4.4, Table 32-1"
        SYNTAX  INTEGER {
            cnlNone         (1),
            cnlAdmin        (2),
            cnlComponent    (3)
        }



    -- ****************************************************************
    -- The Global Table. This group will contain the MIB objects that
    -- apply to the whole bridge component or end station.
    -- ****************************************************************

    ieee8021CnGlobalTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021CnGlobalEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A table containing the global variables for each component of
             a bridge or for an end station.  A value of 1 is used in a
             bridge or end station that does not have multiple components.

             The contents of this table SHALL be maintained across a restart
             of the system.
            "
        REFERENCE
            "802.1Qau clause 12.17.1"
        ::= { ieee8021CnMIBObjects 1 }

    ieee8021CnGlobalEntry OBJECT-TYPE
        SYNTAX      Ieee8021CnGlobalEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "A list of objects pertaining to a whole bridge component or
             end station.
            "
        INDEX { ieee8021CnGlobalComponentId }
        ::= { ieee8021CnGlobalTable 1 }

    Ieee8021CnGlobalEntry ::= SEQUENCE {
            ieee8021CnGlobalComponentId  IEEE8021PbbComponentIdentifier,
            ieee8021CnGlobalMasterEnable            TruthValue,
            ieee8021CnGlobalCnmTransmitPriority     IEEE8021PriorityValue,
            ieee8021CnGlobalDiscardedFrames         Counter64
        }

    ieee8021CnGlobalComponentId OBJECT-TYPE
        SYNTAX      IEEE8021PbbComponentIdentifier
        MAX-ACCESS  not-accessible
```

```
        STATUS      current
        DESCRIPTION
           "The bridge component within the system to which the information
            in this ieee8021CnGlobalEntry applies.  If the system is not
            a Bridge, or if only one component is present in the Bridge,
            then this variable (index) MUST be equal to 1.
           "
        ::= { ieee8021CnGlobalEntry 1 }

    ieee8021CnGlobalMasterEnable OBJECT-TYPE
        SYNTAX      TruthValue
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "The state of the congestion notification feature on this bridge
            component or system.  If true, Congestion notification is
            enabled, and if false, congestion notification is disabled.
           "
        REFERENCE
           "802.1Qau clause 32.2.1"
        ::= { ieee8021CnGlobalEntry 2 }

    ieee8021CnGlobalCnmTransmitPriority OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "The priority to use for all Congestion Notification Messages
            transmitted by this bridge component or end station.
           "
        REFERENCE
           "802.1Qau clause 32.2.2"
        ::= { ieee8021CnGlobalEntry 3 }

    ieee8021CnGlobalDiscardedFrames OBJECT-TYPE
        SYNTAX      Counter64
        UNITS       "frames"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The number of frames discarded from full CP queues, in spite
            of the efforts of congestion notification to avoid discards.

            This object is incremented whenever any of the
            ieee8021CnCpDiscardedFrames objects on any Port or priority in
            this same component are incremented.

            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            ifCounterDiscontinuityTime object of the associated
            interface (if any).
           "
        REFERENCE
```

```
           "802.1Qau clause 32.2.3"
       ::= { ieee8021CnGlobalEntry 4 }


    -- ****************************************************************
    -- The CN Errored Port Table.  One per bridge component or end station.
    -- Scanning this table reveals which Interfaces at which priorities
    -- have an Alternate Priority value that is a
    -- Congestion Notification Priority Value.
    -- ****************************************************************

    ieee8021CnErroredPortTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021CnErroredPortEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "There is one Errored Port Table per bridge component or end
            station. It permits the retrieval of information about which
            interfaces have congestion notification configuration errors,
            namely, those specifying an alternate priority that is a CNPV.
           "
        REFERENCE
           "802.1Qau clause 32.2.4"
        ::= { ieee8021CnMIBObjects 2 }

    ieee8021CnErroredPortEntry OBJECT-TYPE
        SYNTAX      Ieee8021CnErroredPortEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "A list of interfaces whose ieee8021CnComPriAlternatePriority
            and/or ieee8021CnPortPriAlternatePriority specify a priority
            value that is a Congestion Notification Priority Value.
           "
        REFERENCE
           "802.1Qau clause 32.2.4"
        INDEX { ieee8021CnEpComponentId,
                ieee8021CnEpPriority,
                ieee8021CnEpIfIndex}
        ::= { ieee8021CnErroredPortTable 1 }

    Ieee8021CnErroredPortEntry ::= SEQUENCE {
            ieee8021CnEpComponentId     IEEE8021PbbComponentIdentifier,
            ieee8021CnEpPriority        IEEE8021PriorityValue,
            ieee8021CnEpIfIndex         InterfaceIndex
        }

    ieee8021CnEpComponentId OBJECT-TYPE
        SYNTAX      IEEE8021PbbComponentIdentifier
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "The bridge component within the system to which the information
```

```
            in this ieee8021CnErroredPortEntry applies.  If the system is
            not a Bridge, or if only one component is present in the
            Bridge, then this variable (index) MUST be equal to 1.
           "
        REFERENCE
           "802.1Qau clause 32.2.4"
        ::= { ieee8021CnErroredPortEntry 1 }

    ieee8021CnEpPriority OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "The priority value whose alternate priority is misconfigured.
           "
        REFERENCE
           "802.1Qau clause 32.2.4"
        ::= { ieee8021CnErroredPortEntry 2 }

    ieee8021CnEpIfIndex OBJECT-TYPE
        SYNTAX      InterfaceIndex
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "This object represents the Bridge Port or aggregated port
            on which the congestion notification alternate priority is
            misconfigured.
            Upon a restart of the system, the system SHALL, if necessary,
            change the value of this variable so that it references the row
            in the ifXTable with the same value of ifAlias that it
            referenced before the system restart.  If no such row exists,
            then the system SHALL delete this row in the
            ieee8021CnErroredPortTable.
           "
        REFERENCE
           "802.1Qau clause 32.2.4"
        ::= { ieee8021CnErroredPortEntry 3 }


    -- ****************************************************************
    -- The CN Per-component per-priority table.  One table per bridge
    -- component or end station, one entry per
    -- Congestion Notification Priority Value
    -- ****************************************************************

    ieee8021CnCompntPriTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021CnCompntPriEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "Each row in this table supplies default values for one
            Congestion Notification Priority Value for a whole bridge
            component or end station.

```

```
            Creating a row in this table makes the priority value of
            ieee8021CnComPriPriority a
            Congestion Notification Priority Value.
            Deleting a row in this table makes the value in the deleted
            ieee8021CnComPriPriority no longer a
            Congestion Notification Priority Value.

            A  system SHALL NOT allow eight rows in this table
            to be created with the same value of
            ieee8021CnComPriComponentId; see the description of
            ieee8021CnComPriRowStatus.

            The contents of this table SHALL be maintained across a restart
            of the system.
           "
        REFERENCE
           "802.1Qau clause 12.17.2, 12.17.2.1, 12.17.2.2"
        ::= { ieee8021CnMIBObjects 3 }

    ieee8021CnCompntPriEntry OBJECT-TYPE
        SYNTAX      Ieee8021CnCompntPriEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "One entry per Congestion Notification Priority Value per
            bridge component or end station.
           "
        REFERENCE
           "802.1Qau clause 12.17.2, 12.17.2.1, 12.17.2.2"
        INDEX { ieee8021CnComPriComponentId,
                ieee8021CnComPriPriority }
        ::= { ieee8021CnCompntPriTable 1 }

    Ieee8021CnCompntPriEntry ::= SEQUENCE {
        ieee8021CnComPriComponentId       IEEE8021PbbComponentIdentifier,
        ieee8021CnComPriPriority          IEEE8021PriorityValue,
        ieee8021CnComPriDefModeChoice     Ieee8021CnControlChoice,
        ieee8021CnComPriAlternatePriority IEEE8021PriorityValue,
        ieee8021CnComPriAutoAltPri        IEEE8021PriorityValue,
        ieee8021CnComPriAdminDefenseMode  Ieee8021CnDefenseMode,
        ieee8021CnComPriCreation          INTEGER,
        ieee8021CnComPriLldpInstanceChoice    Ieee8021CnLldpChoice,
        ieee8021CnComPriLldpInstanceSelector  LldpV2DestAddressTableIndex,
        ieee8021CnComPriRowStatus         RowStatus
        }

    ieee8021CnComPriComponentId OBJECT-TYPE
        SYNTAX      IEEE8021PbbComponentIdentifier
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "The bridge component within the system to which the information
            in this ieee8021CnCompntPriEntry applies.  If the system is
            not a Bridge, or if only one component is present in the
```

```
            Bridge, then this variable (index) MUST be equal to 1.
           "
        REFERENCE
           "802.1Qau clause 12.17.2, 12.17.2.1, 12.17.2.2"
        ::= { ieee8021CnCompntPriEntry 1 }

    ieee8021CnComPriPriority OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "The Congestion Notification Priority Value for which this
            row supplies default values.
           "
        REFERENCE
           "802.1Qau clauses 12.17.2"
        ::= { ieee8021CnCompntPriEntry 2 }

    ieee8021CnComPriDefModeChoice OBJECT-TYPE
        SYNTAX      Ieee8021CnControlChoice {
                        cpcAdmin    (1),
                        cpcAuto     (2)
                    }
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
           "Specifies how the default CND defense mode and alternate
            priority for this Congestion Notification Priority Value on all
            ports on this bridge component or end station are to be chosen,
            either:

                cpcAdmin(1) Default CND defense mode is chosen by
                            ieee8021CnComPriAdminDefenseMode, and alternate
                            priority by ieee8021CnComPriAlternatePriority.
                cpcAuto(2)  Default CND defense mode is chosen by
                            ieee8021CnPortPriAutoDefenseMode, and alternate
                            priority by ieee8021CnComPriAutoAltPri.

            This variable can be overridden by
            ieee8021CnPortPriDefModeChoice.
           "
        REFERENCE
           "802.1Qau clause 32.1.3, 32.3.1"
        DEFVAL { cpcAuto }
        ::= { ieee8021CnCompntPriEntry 3 }

    ieee8021CnComPriAlternatePriority OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
           "The Congestion Notification Priority Value to which an
            incoming frame is to be mapped, in spite of what the
            Priority Regereration Table says, if 1) Congestion
```

```
            Notification is enabled and 2) the CND defense mode of the
            port is cptEdge.

            Deleting a row in this table does not alter the value of any
            other row's ieee8021CnComPriAlternatePriority.
           "
        REFERENCE
           "802.1Qau clauses 32.3.2"
        DEFVAL { 0 }
        ::= { ieee8021CnCompntPriEntry 4 }

    ieee8021CnComPriAutoAltPri OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The Congestion Notification Priority Value to which an
            incoming frame can be mapped, in spite of what the
            Priority Regereration Table says, if 1) Congestion
            Notification is enabled, 2) the CND defense mode of the
            port is cptEdge, and 3) ieee8021CnComPriDefModeChoice
            contains the value cpcAuto (2).

            The value of this object is the next lower priority value
            than this row's ieee8021CnComPriPriority that is not a CNPV,
            or the next higher non-CNPV, if all lower values are CNPVs.

            The value of this object, and any consequent priority
            regeneration, is automatically updated by the managed system
            whenever a row in the ieee8021CnCompntPriTable is created or
            deleted.  The value of this object is not dependent upon
            whether congestion notification is enabled or disabled for any
            priority or for the whole bridge component or end station; it
            depends only upon whether the ieee8021CnCompntPriTable row
            exists.
           "
        REFERENCE
           "802.1Qau clauses 32.3.3"
        ::= { ieee8021CnCompntPriEntry 5 }

    ieee8021CnComPriAdminDefenseMode OBJECT-TYPE
        SYNTAX      Ieee8021CnDefenseMode
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
           "The default CND defense mode for this
            Congestion Notification Priority Value on all ports on this
            bridge component or end station.

            This variable can be overridden by
            ieee8021CnPortPriAdminDefenseMode.
           "
        REFERENCE
           "802.1Qau clause 32.1.3, 32.3.4"
```

```
        DEFVAL { cptInterior }
        ::= { ieee8021CnCompntPriEntry 6 }

    ieee8021CnComPriCreation OBJECT-TYPE
        SYNTAX      INTEGER {
                        cncpAutoEnable   (1),
                        cncpAutoDisable  (2)
                    }
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
           "The default value for ieee8021CnComPriDefModeChoice for
            newly-created entries in the ieee8021CnPortPriTable:

                cncpAutoEnable (1)  Newly-created
                                    ieee8021CnPortPriDefModeChoice
                                    objects take the value cpcComp (3).
                cncpAutoDisable(2)  Newly-created
                                    ieee8021CnPortPriDefModeChoice
                                    objects take the value cpcAdmin (1).
           "
        REFERENCE
           "802.1Qau clause 32.3.5"
        DEFVAL { cncpAutoEnable }
        ::= { ieee8021CnCompntPriEntry 7 }

    ieee8021CnComPriLldpInstanceChoice OBJECT-TYPE
        SYNTAX      Ieee8021CnLldpChoice {
                          cnlNone  (1),
                          cnlAdmin (2)
                    }
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
           "Specifies whether or not the default value for all Ports is to
            send and receive the Congestion Notification TLV in LLDPDUs,
            either:
                cnlNone(1)  Do not send Congestion Notification TLVs, and
                            ignore them on receipt.
                cnlAdmin(2) Use the LLDP instance selected by
                            ieee8021CnComPriLldpInstanceSelector to send and
                            receive the Congestion Notification TLV.

            This object can be overridden by
            ieee8021CnPortPriLldpInstanceChoice.
           "
        REFERENCE
           "802.1Qau clause 32.1.3, 32.3.6"
        DEFVAL { cnlAdmin }
        ::= { ieee8021CnCompntPriEntry 8 }

    ieee8021CnComPriLldpInstanceSelector OBJECT-TYPE
        SYNTAX      LldpV2DestAddressTableIndex
        MAX-ACCESS  read-create
```

```
        STATUS      current
        DESCRIPTION
           "Specifies a default value for which LLDP instance is to be
            used to provide the information for automatic configuration
            of ports' CND defense modes (ieee8021CnPortPriAutoDefenseMode).

            This object is ignored by the managed system if
            ieee8021CnComPriLldpInstanceChoice contains the value cnlNone
            (1).

            This object can be overridden by
            ieee8021CnPortPriLldpInstanceChoice and
            ieee8021CnPortPriLldpInstanceChoice.
           "
        REFERENCE
           "802.1Qau clause 32.1.3, 32.3.7"
        DEFVAL { 1 }
        ::= { ieee8021CnCompntPriEntry 9 }

    ieee8021CnComPriRowStatus OBJECT-TYPE
        SYNTAX      RowStatus
        MAX-ACCESS  read-create
        STATUS      current
        DESCRIPTION
           "This object indicates the status of an entry, and is used
            to create/delete entries.

            A system SHALL NOT permit eight ieee8021CnComPriRowStatus
            objects, all with the same value of ieee8021CnComPriComponentId,
            to have the value active(1).  An attempt to create or activate
            a row when there are already seven active rows SHALL result in
            that eighth row's ieee8021CnComPriRowStatus having the value
            notReady(3), and the return of an inconsistentValue error.
           "
        REFERENCE
           "802.1Qau clause 30.4"
        ::= { ieee8021CnCompntPriEntry 10 }


    -- ****************************************************************
    -- The CN Per-component per-interface per-priority table.  One table
    -- per end station or bridge component.  One entry per interface per
    -- priority value, but only for priority values that are
    -- Congestion Notification Priority Values (CNPVs).  Controls
    -- a CNPV on a specific port when the default values in
    -- ieee8021CnCompntPriTable need to be overridden.
    -- ****************************************************************

    ieee8021CnPortPriTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021CnPortPriEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "Each row in this table supplies values for one port's
```

```
            Congestion Notification Priority Value (CNPV).

            Creating an entry in ieee8021CnCompntPriTable creates this
            entry, with the default values, on all ports in the bridge
            component or end station.  Deleting an entry in
            ieee8021CnCompntPriTable deletes this ieee8021CnCompntPriEntry
            on all ports in the bridge component or end station.

            The contents of this table SHALL be maintained across a restart
            of the system, except as noted in the description of
            ieee8021CnPortPriIfIndex.
           "
        REFERENCE
           "802.1Qau clause 12.17.3"
        ::= { ieee8021CnMIBObjects 4 }

    ieee8021CnPortPriEntry OBJECT-TYPE
        SYNTAX      Ieee8021CnPortPriEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "One entry per port per Congestion Notification Priority Value
            per bridge component or end station.
           "
        REFERENCE
           "802.1Qau clause 12.17.3"
        INDEX { ieee8021CnPortPriComponentId,
                ieee8021CnPortPriority,
                ieee8021CnPortPriIfIndex }
        ::= { ieee8021CnPortPriTable 1 }

    Ieee8021CnPortPriEntry ::= SEQUENCE {
            ieee8021CnPortPriComponentId    IEEE8021PbbComponentIdentifier,
            ieee8021CnPortPriority              IEEE8021PriorityValue,
            ieee8021CnPortPriIfIndex            InterfaceIndex,
            ieee8021CnPortPriDefModeChoice      Ieee8021CnControlChoice,
            ieee8021CnPortPriAdminDefenseMode   Ieee8021CnDefenseMode,
            ieee8021CnPortPriAutoDefenseMode    Ieee8021CnDefenseMode,
            ieee8021CnPortPriLldpInstanceChoice Ieee8021CnLldpChoice,
            ieee8021CnPortPriLldpInstanceSelector
                                        LldpV2DestAddressTableIndex,
            ieee8021CnPortPriAlternatePriority  IEEE8021PriorityValue
        }

    ieee8021CnPortPriComponentId OBJECT-TYPE
        SYNTAX      IEEE8021PbbComponentIdentifier
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "The bridge component within the system to which the information
            in this ieee8021CnPortPriEntry applies.  If the system is
            not a Bridge, or if only one component is present in the
            Bridge, then this variable (index) MUST be equal to 1.
           "
```

```
        REFERENCE
           "802.1Qau clause 12.17.3"
        ::= { ieee8021CnPortPriEntry 1 }

    ieee8021CnPortPriority OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "The Congestion Notification Priority Value for which
            this row supplies default values.
           "
        REFERENCE
           "802.1Qau clauses 12.17.3"
        ::= { ieee8021CnPortPriEntry 2 }

    ieee8021CnPortPriIfIndex OBJECT-TYPE
        SYNTAX      InterfaceIndex
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "This object represents the port or aggregated port
            to which the entry applies.

            Upon a restart of the system, the system SHALL, if necessary,
            change the value of this object, and rearrange the order of the
            ieee8021CnPortPriTable, so that the value in
            ieee8021CnPortPriIfIndex references the row in the ifXTable
            with the same value for ifAlias that it referenced before the
            system restart.  If no such entry exists in the ifXTable, then
            the system SHALL delete the row in the ieee8021CnPortPriTable.
           "
        REFERENCE
           "802.1Qau clause 12.17.3"
        ::= { ieee8021CnPortPriEntry 3 }

    ieee8021CnPortPriDefModeChoice OBJECT-TYPE
        SYNTAX  Ieee8021CnControlChoice
        MAX-ACCESS  read-write
        STATUS current
        DESCRIPTION
           "This object determines how the CND defense mode and alternate
            priority value of this port for this CNPV is to be selected,
            either:

                cpcAdmin(1) CND defense mode is controlled by
                            ieee8021CnPortPriAdminDefenseMode, and alternate
                            priority by ieee8021CnPortPriAlternatePriority.
                cpcAuto(2)  CND defense mode is controlled by
                            ieee8021CnPortPriAutoDefenseMode and alternate
                            priority by ieee8021CnComPriAlternatePriority.
                cpcComp(3)  CND defense mode and alternate priority are
                            controlled by
                            ieee8021CnComPriDefModeChoice.
```

```
1
2               This variable can override ieee8021CnComPriDefModeChoice.
3              "
4           REFERENCE
5              "IEEE 802.1Qau clause 32.1.3, 32.4.1"
6           DEFVAL { cpcComp }
7           ::= { ieee8021CnPortPriEntry 4 }
8
9       ieee8021CnPortPriAdminDefenseMode OBJECT-TYPE
10          SYNTAX  Ieee8021CnDefenseMode
11          MAX-ACCESS  read-write
12          STATUS current
13          DESCRIPTION
14             "This object indicates the operator's choice for the CND defense
15              mode in which this port is to operate for this CNPV whenever
16              ieee8021CnPortPriDefModeChoice has the value cpcAdmin(1).
17
18              This variable can override ieee8021CnComPriDefModeChoice.
19             "
20          REFERENCE
21             "IEEE 802.1Qau clause 32.1.3, 32.4.1"
22          DEFVAL { cptDisabled }
23          ::= { ieee8021CnPortPriEntry 5 }
24
25      ieee8021CnPortPriAutoDefenseMode OBJECT-TYPE
26          SYNTAX  Ieee8021CnDefenseMode {
27                      cptInterior      (2),
28                      cptInteriorReady (3),
29                      cptEdge          (4)
30                  }
31          MAX-ACCESS read-only
32          STATUS current
33          DESCRIPTION
34             "This object indicates in which the CND defense mode this port
35              would operate for this CNPV as determined by the LLDP
36              Congestion Notification TLV.
37             "
38          REFERENCE
39             "IEEE 802.1Qau clause 32.4.3"
40          ::= { ieee8021CnPortPriEntry 6 }
41
42
43      ieee8021CnPortPriLldpInstanceChoice OBJECT-TYPE
44          SYNTAX      Ieee8021CnLldpChoice
45          MAX-ACCESS  read-write
46          STATUS      current
47          DESCRIPTION
48             "Specifies how to determine the LLDP instance to be used for the
49              Congestion Notification TLV, either:
50                  cnlNone(1)      No LLDP Congestion Notification TLV is to
51                                  carry Per-priority CNPV indicators or
52                                  Per-priority Ready indicators on this Port
53                                  for this priority.
54                  cnlAdmin(2)     ieee8021CnPortPriLldpInstanceSelector
```

```
                                governs which LLDP instance is to carry
                                Per-priority CNPV indicators and
                                Per-priority Ready indicators for this
                                priority in its Congestion Notification TLV
                                on this Port
                cnlComponent(3) ieee8021CnComPriLldpInstanceChoice
                                governs LLDP instance selection for this
                                Port and priority.

             This object can override ieee8021CnComPriLldpInstanceChoice.
           "
        REFERENCE
           "802.1Qau clause 32.1.3, 32.4.4"
        DEFVAL { cnlComponent }
        ::= { ieee8021CnPortPriEntry 7 }

    ieee8021CnPortPriLldpInstanceSelector OBJECT-TYPE
        SYNTAX      LldpV2DestAddressTableIndex
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object determines which LLDP instance selector, if any,
            is used for automatic determination of the CND defense mode for
            this port and CNPV.

            This object can override ieee8021CnComPriLldpInstanceSelector.
           "
        REFERENCE
           "802.1Qau clause 32.1.3, 32.4.5"
        DEFVAL { 3 }
        ::= { ieee8021CnPortPriEntry 8 }

    ieee8021CnPortPriAlternatePriority OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "The Congestion Notification Priority Value to which an
            incoming frame is to be mapped, in spite of what the
            Priority Regereration Table says, if 1) Congestion
            Notification is enabled and 2) the port is acting in the
            cptEdge (4) CND defense mode.

            This object is ignored unless ieee8021CnPortPriDefModeChoice
            contains the value cpcAdmin (1).
           "
        REFERENCE
           "802.1Qau clause 32.4.6"
        DEFVAL { 0 }
        ::= { ieee8021CnPortPriEntry 9 }


    -- ****************************************************************
```

```
-- The CN per-CP table.  One table per end station or bridge component.
-- One entry per Congestion Point.  An entry in this table controls one
-- Congestion Point (CP).
-- ********************************************************************

ieee8021CnCpTable OBJECT-TYPE
    SYNTAX      SEQUENCE OF Ieee8021CnCpEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
       "Each row in this table supplies values for one
        Congestion Point (CP).

        This table is indexed by component, port (interface), and
        an arbitrary CP index.  This arbitrary CP index is not
        necessarily the Congestion Point Identifier (CPID) carried in
        Congestion Notification Messages (CNMs).

        Creating an entry in ieee8021CnCompntPriTable can create an
        entry in this table, with the default values, on all ports in
        the bridge component or end station.  Because more than one
        Congestion Notification Priority Value (CNPV) can flow
        through a single CP, the creation of an entry in
        ieee8021CnCompntPriTable does not necessarily create a new
        entry in this table.  An end station can have more than one
        CP for the same CNPV, so creating an entry in
        ieee8021CnCompntPriTable can create multiple entries in this
        table.

        Because each port in a bridge component or end station can have
        a different relationship between CNPVs and CPs, the entries
        created or deleted on each port can be different.

        Deleting the last entry in ieee8021CnCompntPriTable for a
        CNPV passing through the CP controlled by this entry deletes
        the entry on some or all of the ports in the bridge component
        or end station.

        Because each port in a bridge component or end station can have
        a different relationship between CNPVs and CPs, the entries
        created or deleted on each port can be different.

        The relationship between ieee8021CnCpIndex
        values and CPs is an implementation dependent matter.

        The contents of this table SHALL be maintained across a restart
        of the system, except as noted in the description of
        ieee8021CnCpIfIndex.
       "
    REFERENCE
       "802.1Qau clause 12.17.4"
    ::= { ieee8021CnMIBObjects 5 }

ieee8021CnCpEntry OBJECT-TYPE
```

```
        SYNTAX      Ieee8021CnCpEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "An entry in the Congestion Point table controls a single
            Congestion Point on a port in a bridge component or end station.
           "
        REFERENCE
           "802.1Qau clause 12.17.4"
        INDEX { ieee8021CnCpComponentId,
                ieee8021CnCpIfIndex,
                ieee8021CnCpIndex }
        ::= { ieee8021CnCpTable 1 }

    Ieee8021CnCpEntry ::= SEQUENCE {
            ieee8021CnCpComponentId      IEEE8021PbbComponentIdentifier,
            ieee8021CnCpIfIndex              InterfaceIndex,
            ieee8021CnCpIndex                Unsigned32,
            ieee8021CnCpPriority             IEEE8021PriorityValue,
            ieee8021CnCpMacAddress           MacAddress,
            ieee8021CnCpIdentifier           OCTET STRING,
            ieee8021CnCpQueueSizeSetPoint    Unsigned32,
            ieee8021CnCpFeedbackWeight       Integer32,
            ieee8021CnCpMinSampleBase        Unsigned32,
            ieee8021CnCpDiscardedFrames      Counter64,
            ieee8021CnCpTransmittedFrames    Counter64,
            ieee8021CnCpTransmittedCnms      Counter64,
            ieee8021CnCpMinHeaderOctets      Unsigned32
        }

    ieee8021CnCpComponentId OBJECT-TYPE
        SYNTAX      IEEE8021PbbComponentIdentifier
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "The bridge component within the system to which the information
            in this ieee8021CnCpEntry applies.  If the system is
            not a Bridge, or if only one component is present in the
            Bridge, then this variable (index) MUST be equal to 1.
           "
        REFERENCE
           "802.1Qau clause 12.17.4"
        ::= { ieee8021CnCpEntry 1 }

    ieee8021CnCpIfIndex OBJECT-TYPE
        SYNTAX      InterfaceIndex
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "This object represents the port or aggregated port
            to which the entry applies.

            Upon a restart of the system, the system SHALL, if necessary,
            change the value of this object, and rearrange the order of the
```

```
            ieee8021CnCpTable, so that the value in ieee8021CnCpIfIndex
            references the row in the ifXTable with the same value for
            ifAlias that it referenced before the system restart.  If no
            such entry exists in the ifXTable, then the system SHALL delete
            the row in the ieee8021CnCpTable.
          "
        REFERENCE
          "802.1Qau clause 12.17.4"
        ::= { ieee8021CnCpEntry 2 }

    ieee8021CnCpIndex OBJECT-TYPE
        SYNTAX      Unsigned32 (1..4096)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
          "This object is an arbitrary integer indexing the entries in
           this table among the entries for the same component and
           interface.  In a system that supports no more than one
           Congestion Point per priority per interface, ieee8021CnCpIndex
           SHALL be equal to the lowest numerical
           Congestion Notification Priority Value served by this
           Congestion Point.  Otherwise, it SHOULD be a small integer
           value.
          "
        REFERENCE
          "802.1Qau clause 12.17.4"
        ::= { ieee8021CnCpEntry 3 }

    ieee8021CnCpPriority OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
          "This object indicates the lowest numerical
           Congestion Notification Priority Value that this entry's
           Congestion Point serves.
          "
        REFERENCE
          "802.1Qau clause 12.17.4"
        ::= { ieee8021CnCpEntry 4 }

    ieee8021CnCpMacAddress OBJECT-TYPE
        SYNTAX      MacAddress
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
          "This object indicates the MAC address used as the source
           address in Congestion Notification Message transmitted
           by this Congestion Point.
          "
        REFERENCE
          "802.1Qau clause 32.8.1"
        ::= { ieee8021CnCpEntry 5 }
```

```
ieee8021CnCpIdentifier OBJECT-TYPE
    SYNTAX      OCTET STRING (SIZE(8))
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
       "This object indicates the Congestion Point Identifier (CPID)
        transmitted in Congestion Notification Message by this
        Congestion Point.

        It is not specified whether the CPID reported in a CNM by a CP
        that serves multiple CNPVs does or does not have the same value
        for its different CNPVs.
       "
    REFERENCE
       "802.1Qau clause 32.8.2"
    ::= { ieee8021CnCpEntry 6 }

ieee8021CnCpQueueSizeSetPoint OBJECT-TYPE
    SYNTAX      Unsigned32 (100..4294967295)
    UNITS       "octets"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "This object is the set point for the queue managed by this
        Congestion Point (CP).  Congestion Notification Messages are
        transmitted to the sources of frames queued in this CP's
        queue in order to keep the total number of octets stored in
        the queue at this set point.
       "
    REFERENCE
       "802.1Qau clause 30.2, 32.8.3"
    DEFVAL { 26000 }
    ::= { ieee8021CnCpEntry 7 }

ieee8021CnCpFeedbackWeight OBJECT-TYPE
    SYNTAX      Integer32 (-10..10)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "This object controls the weight (cpW) change in queue length
        in the calculation of cpFb when the Congestion Point is
        generating a Congestion Notification Message.

        The weight cpW is equal to two to the power of this object.
        Thus, if this object contains a -1, cpW = 1/2.
       "
    REFERENCE
       "802.1Qau clause 32.8.6"
    DEFVAL { 1 }
    ::= { ieee8021CnCpEntry 8 }

ieee8021CnCpMinSampleBase OBJECT-TYPE
    SYNTAX      Unsigned32 (10000..4294967295)
    UNITS       "octets"
```

```
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object determines the minimum number of octets to
            enqueue in the Congestion Point's queue between transmissions
            of Congestion Notification Messages.
           "
        REFERENCE
           "802.1Qau clause 32.8.11"
        DEFVAL { 150000 }
        ::= { ieee8021CnCpEntry 9 }

    ieee8021CnCpDiscardedFrames OBJECT-TYPE
        SYNTAX      Counter64
        UNITS       "frames"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The number of data frames discarded by the queue controlled
            by this Congestion Point due to queue congestion.

            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            ifCounterDiscontinuityTime object of the associated
            interface (if any).
           "
        REFERENCE
           "802.1Qau clause 32.8.12"
        ::= { ieee8021CnCpEntry 10 }

    ieee8021CnCpTransmittedFrames OBJECT-TYPE
        SYNTAX      Counter64
        UNITS       "frames"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The number of data frames passed on to the queue controlled by
            this Congestion Point that were not discarded due to queue
            congestion.

            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            ifCounterDiscontinuityTime object of the associated
            interface (if any).
           "
        REFERENCE
           "802.1Qau clause 32.8.13"
        ::= { ieee8021CnCpEntry 11 }

    ieee8021CnCpTransmittedCnms OBJECT-TYPE
        SYNTAX      Counter64
        UNITS       "frames"
```

```
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The number of Congestion Notification Message transmitted
            by this Congestion Point.

            Discontinuities in the value of this counter can occur
            at re-initialization of the management system, and at
            other times as indicated by the value of
            ifCounterDiscontinuityTime object of the associated
            interface (if any).
           "
        REFERENCE
           "802.1Qau clause 32.8.14"
        ::= { ieee8021CnCpEntry 12 }

    ieee8021CnCpMinHeaderOctets OBJECT-TYPE
        SYNTAX      Unsigned32 (0..64)
        UNITS       "octets"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "Specifies the minimum number of octets to be returned in a
            Congestion Notification Message from the mac_service_data_unit
            of the data frame that triggered transmission of the CNM.  If
            the mac_service_data_unit has fewer octets than the value of
            this object, then all of the mac_service_data_unit is returned
            in the CNM.
           "
        REFERENCE
           "802.1Qau clause 32.8.15, 32.9.4:k"
        DEFVAL { 0 }
        ::= { ieee8021CnCpEntry 13 }



    -- ****************************************************************
    -- The CPID to {Interface, CP index} table.  One table per system.
    -- One entry per CPID.
    -- ****************************************************************

    ieee8021CnCpidToInterfaceTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021CnCpidToInterfaceEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "This table allows the network manager to obtain the
            interface index and CP index needed to access an entry in
            the ieee8021CnCpTable, given a Congestion Point Identifier
            (CPID) received a Congestion Notification Messages (CNMs).

            Upon a restart of the system, the system SHALL, if necessary,
            update this table to be consistent with the ieee8021CnCpTable.
           "
```

```
        REFERENCE
           "802.1Qau clause 17.2.16, 12.17.4, 32.8.2"
        ::= { ieee8021CnMIBObjects 6 }

    ieee8021CnCpidToInterfaceEntry OBJECT-TYPE
        SYNTAX      Ieee8021CnCpidToInterfaceEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "An entry in the ieee8021CnCpidToInterfaceTable.  Translates
            a Congestion Point Identifier to a component identifier,
            interface index, and CP index
           "
        REFERENCE
           "802.1Qau clause 17.2.16"
        INDEX { ieee8021CnCpidToIfCpid }
        ::= { ieee8021CnCpidToInterfaceTable 1 }

    Ieee8021CnCpidToInterfaceEntry ::= SEQUENCE {
            ieee8021CnCpidToIfCpid           OCTET STRING,
            ieee8021CnCpidToIfComponentId    IEEE8021PbbComponentIdentifier,
            ieee8021CnCpidToIfIfIndex        InterfaceIndex,
            ieee8021CnCpidToIfCpIndex        Unsigned32
        }

    ieee8021CnCpidToIfCpid OBJECT-TYPE
        SYNTAX      OCTET STRING (SIZE(8))
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "This object is the Congestion Point Identifier (CPID)
            transmitted in Congestion Notification Message by a
            Congestion Point residing in this bridge component or
            end station.
           "
        REFERENCE
           "802.1Qau clause 17.2.16, 32.8.2"
        ::= { ieee8021CnCpidToInterfaceEntry 1 }

    ieee8021CnCpidToIfComponentId OBJECT-TYPE
        SYNTAX      IEEE8021PbbComponentIdentifier
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "The bridge component within the system to which the information
            in this ieee8021CnCpidToInterfaceEntry applies.  If the system
            is not a Bridge, or if only one component is present in the
            Bridge, then this variable (index) MUST be equal to 1.
           "
        REFERENCE
           "802.1Qau clause 17.2.16"
        ::= { ieee8021CnCpidToInterfaceEntry 2 }

    ieee8021CnCpidToIfIfIndex OBJECT-TYPE
```

```
        SYNTAX      InterfaceIndex
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "This object indicates the interface on which the selected
            Congestion Point resides.  This value can be used, along
            with ieee8021CnCpidToIfCpIndex, to find the Congestion Point
            in the ieee8021CnCpTable.
           "
        REFERENCE
           "802.1Qau clause 17.2.16"
        ::= { ieee8021CnCpidToInterfaceEntry 3 }

    ieee8021CnCpidToIfCpIndex OBJECT-TYPE
        SYNTAX      Unsigned32 (1..4096)
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "This object indicates the Congestion Point's index on the
            interface on which the selected Congestion Point resides.
            This value can be used, along with ieee8021CnCpidToIfIfIndex,
            to find the Congestion Point in the ieee8021CnCpTable.
           "
        REFERENCE
           "802.1Qau clause 17.2.16"
        ::= { ieee8021CnCpidToInterfaceEntry 4 }



    -- ********************************************************************
    -- The Reaction Point Port table.  One table per end station.  One
    -- entry per Port per CNPV.
    -- ********************************************************************

    ieee8021CnRpPortPriTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021CnRpPortPriEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "Each row in this table supplies values for all of the
            Reaction Points (RPs) on one Port and one priority of an end
            station or bridge component.  This table is indexed by
            component, port (interface), and priority.

            Creating an entry in ieee8021CnCompntPriTable can create an
            entry in this table, with the default values, on all ports
            in the end station.

            Deleting the an entry in ieee8021CnCompntPriTable for a
            CNPV passing through the RP controlled by this entry deletes
            entries on some or all of the ports in the end station.

            The contents of this table SHALL be maintained across a restart
            of the system, except as noted in the description of
```

```
            ieee8021CnRpPortPriIfIndex.
          "
      REFERENCE
          "802.1Qau clause 12.17.5"
      ::= { ieee8021CnMIBObjects 7 }

  ieee8021CnRpPortPriEntry OBJECT-TYPE
      SYNTAX      Ieee8021CnRpPortPriEntry
      MAX-ACCESS  not-accessible
      STATUS      current
      DESCRIPTION
          "An entry in the Reaction Point table controls all of the
           Reaction Points on a port in an end station that share the same
           priority value.
          "
      REFERENCE
          "802.1Qau clause 12.17.5"
      INDEX { ieee8021CnRpPortPriComponentId, ieee8021CnRpPortPriPriority,
              ieee8021CnRpPortPriIfIndex
            }
      ::= { ieee8021CnRpPortPriTable 1 }

  Ieee8021CnRpPortPriEntry ::= SEQUENCE {
          ieee8021CnRpPortPriComponentId   IEEE8021PbbComponentIdentifier,
          ieee8021CnRpPortPriPriority      IEEE8021PriorityValue,
          ieee8021CnRpPortPriIfIndex       InterfaceIndex,
          ieee8021CnRpPortPriMaxRps        Unsigned32,
          ieee8021CnRpPortPriCreatedRps    Counter32,
          ieee8021CnRpPortPriCentiseconds  Counter64
      }

  ieee8021CnRpPortPriComponentId OBJECT-TYPE
      SYNTAX      IEEE8021PbbComponentIdentifier
      MAX-ACCESS  not-accessible
      STATUS      current
      DESCRIPTION
          "The bridge component within the system to which the information
           in this ieee8021CnRpGroupEntry applies.  If the system is
           not a Bridge, or if only one component is present in the
           Bridge, then this variable (index) MUST be equal to 1.
          "
      ::= { ieee8021CnRpPortPriEntry 1 }

  ieee8021CnRpPortPriPriority OBJECT-TYPE
      SYNTAX      IEEE8021PriorityValue
      MAX-ACCESS  not-accessible
      STATUS      current
      DESCRIPTION
          "This object indicates the lowest numerical
           Congestion Notification Priority Value that this entry's
           Reaction Point serves.
          "
      REFERENCE
          "802.1Qau clause 12.17.5"
```

```
        ::= { ieee8021CnRpPortPriEntry 2 }

    ieee8021CnRpPortPriIfIndex OBJECT-TYPE
        SYNTAX      InterfaceIndex
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "This object indicates the interface on which the selected
            Reaction Points reside.

            Upon a restart of the system, the system SHALL, if necessary,
            change the value of this object, and rearrange the order of the
            ieee8021CnRpPortPriTable, so that the value in
            ieee8021CnRpPortPriIfIndex references the row in the ifXTable
            with the same value for ifAlias that it referenced before the
            system restart.  If no such entry exists in the ifXTable, then
            the system SHALL delete the row in the
            ieee8021CnRpPortPriTable.
           "
        REFERENCE
           "802.1Qau clause 12.17.5"
        ::= { ieee8021CnRpPortPriEntry 3 }

    ieee8021CnRpPortPriMaxRps OBJECT-TYPE
        SYNTAX      Unsigned32 (1..100)
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "An integer controlling the maximum number of Reaction Points
            allowed for this CNPV on this Port. An end station SHALL
            not create more than this many Reaction Point on this Port,
            but it MAY create fewer.
           "
        REFERENCE
           "802.1Qau clause 32.10.1"
        DEFVAL { 1 }
        ::= { ieee8021CnRpPortPriEntry 4 }

    ieee8021CnRpPortPriCreatedRps OBJECT-TYPE
        SYNTAX      Counter32
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "This object returns the number of times any of the
            Reaction Points (RPs) controlled by this entry has had
            its rpEnabled variable set TRUE by the reception of a
            Congestion Notification Message.

            Dividing the change in ieee8021CnRpPortPriCentiseconds by the
            change in this object over a time interval yields the average
            lifetime of an active RP during that interval.
           "
        REFERENCE
           "802.1Qau clause 32.10.2, 32.10.3, 32.13.1"
```

```
        ::= { ieee8021CnRpPortPriEntry 5 }

    ieee8021CnRpPortPriCentiseconds OBJECT-TYPE
        SYNTAX      Counter64
        UNITS       "centiseconds"
        MAX-ACCESS  read-only
        STATUS      current
        DESCRIPTION
           "This object returns the total number of centi-seconds that
            any of the Reaction Points (RPs) controlled by this entry
            has had its rpEnabled variable in the TRUE state.  That is,
            once each centi-second, this counter is incremented by the
            number of RPs this entry controls that are actively rate
            limiting output frames.

            Dividing the change in this object over a time interval by the
            length of the interval yields the average number of RPs active
            over that interval.  Dividing the change in this object by the
            change in ieee8021CnRpPortPriCreatedRps over that same time
            interval yields the average lifetime of an active RP during that
            interval.
           "
        REFERENCE
           "802.1Qau clause 32.10.3, 32.13.1"
        ::= { ieee8021CnRpPortPriEntry 6 }



    -- ********************************************************************
    -- The Reaction Point Group table.  One table per end station.
    -- One entry per Reaction Point.
    -- ********************************************************************

    ieee8021CnRpGroupTable OBJECT-TYPE
        SYNTAX      SEQUENCE OF Ieee8021CnRpGroupEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "Each row in this table supplies values for one or more
            Reaction Points (RPs).  This table is indexed by component,
            port (interface), and an arbitrary RP index.

            Creating an entry in ieee8021CnCompntPriTable can create an
            entry in this table, with the default values, on all ports
            in the end station.  An end station can have more than one
            RP for the same Congestion Notification Priority Value
            (CNPV), so creating an entry in ieee8021CnCompntPriTable can
            create multiple entries in this table.

            Because each port in a bridge component or end station can have
            a different relationship between CNPVs and RPs, the entries
            created or deleted on each port can be different.

            Deleting the an entry in ieee8021CnCompntPriTable for a
```

```
            CNPV passing through the RP controlled by this entry deletes
            entries on some or all of the ports in the end station.

            Because each port in an end station can have a
            different relationship between CNPVs and RPs, the entries
            created or deleted on each port can be different.

            The relationship between ieee8021CnRpgIdentifier
            values and RPs is an implementation dependent matter.

            The contents of this table SHALL be maintained across a restart
            of the system, except as noted in the description of
            ieee8021CnRpgIfIndex.
           "
        REFERENCE
           "802.1Qau clause 12.17.6"
        ::= { ieee8021CnMIBObjects 8 }

    ieee8021CnRpGroupEntry OBJECT-TYPE
        SYNTAX      Ieee8021CnRpGroupEntry
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
           "An entry in the Reaction Point table controls a group of
            Reaction Points, on a port in an end station.  All of the
            Reaction Point controlled by this entry serve the same
            Congestion Notification Priority Value.
           "
        INDEX { ieee8021CnRpgComponentId, ieee8021CnRpgPriority,
                ieee8021CnRpgIfIndex, ieee8021CnRpgIdentifier }
        ::= { ieee8021CnRpGroupTable 1 }

    Ieee8021CnRpGroupEntry ::= SEQUENCE {
            ieee8021CnRpgComponentId    IEEE8021PbbComponentIdentifier,
            ieee8021CnRpgPriority       IEEE8021PriorityValue,
            ieee8021CnRpgIfIndex        InterfaceIndex,
            ieee8021CnRpgIdentifier     Unsigned32,
            ieee8021CnRpgEnable         TruthValue,
            ieee8021CnRpgTimeReset      TimeInterval,
            ieee8021CnRpgByteReset      Unsigned32,
            ieee8021CnRpgThreshold      Unsigned32,
            ieee8021CnRpgMaxRate        Unsigned32,
            ieee8021CnRpgAiRate         Unsigned32,
            ieee8021CnRpgHaiRate        Unsigned32,
            ieee8021CnRpgGd             Integer32,
            ieee8021CnRpgMinDecFac      Unsigned32,
            ieee8021CnRpgMinRate        Unsigned32
        }

    ieee8021CnRpgComponentId OBJECT-TYPE
        SYNTAX      IEEE8021PbbComponentIdentifier
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
```

```
            "The bridge component within the system to which the information
             in this ieee8021CnRpGroupEntry applies.  If the system is
             not a Bridge, or if only one component is present in the
             Bridge, then this variable (index) MUST be equal to 1.
            "
        ::= { ieee8021CnRpGroupEntry 1 }

    ieee8021CnRpgPriority OBJECT-TYPE
        SYNTAX      IEEE8021PriorityValue
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This object indicates the lowest numerical
             Congestion Notification Priority Value that this entry's
             Reaction Point serves.
            "
        REFERENCE
            "802.1Qau clause 12.17.5"
        ::= { ieee8021CnRpGroupEntry 2 }

    ieee8021CnRpgIfIndex OBJECT-TYPE
        SYNTAX      InterfaceIndex
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This object indicates the interface on which the group of
             Reaction Points reside.

             Upon a restart of the system, the system SHALL, if necessary,
             change the value of this object, and rearrange the order of the
             ieee8021CnRpGroupTable, so that the value in
             ieee8021CnRpgIfIndex references the row in the ifXTable with
             the same value for ifAlias that it referenced before the system
             restart.  If no such entry exists in the ifXTable, then the
             system SHALL delete the row in the ieee8021CnRpGroupTable.
            "
        REFERENCE
            "802.1Qau clause 12.17.5"
        ::= { ieee8021CnRpGroupEntry 3 }

    ieee8021CnRpgIdentifier OBJECT-TYPE
        SYNTAX      Unsigned32 (1..4096)
        MAX-ACCESS  not-accessible
        STATUS      current
        DESCRIPTION
            "This object is an arbitrary integer indexing the entries in
             this table among the entries for the same interface.  This
             index SHOULD, where possible, be equal to the
             Congestion Notification Priority Value served by this
             Reaction Point.
            "
        REFERENCE
            "802.1Qau clause 12.17.6"
        ::= { ieee8021CnRpGroupEntry 4 }
```

```
ieee8021CnRpgEnable OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "Controls the rpEnabled variable of the Reaction Point state
        machines of the Reaction Points (RPs) controlled by this
        entry as follows:
            true(1)     The rpEnabled variable for the RPs controlled by
                        this object are not held in the FALSE state,
                        thus enabling them to pay attention to received
                        CNMs.
            false(2)    The rpEnabled variable for the RPs controlled by
                        this object are held in the FALSE state, thus
                        disabling them from paying attention to received
                        CNMs.
       "
    REFERENCE
       "802.1Qau clause 32.11.1, 32.13.1"
    DEFVAL { true }
    ::= { ieee8021CnRpGroupEntry 5 }

ieee8021CnRpgTimeReset OBJECT-TYPE
    SYNTAX      TimeInterval
    UNITS       "milliseconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "This object controls the value for all of the state machine
        variables, rpgTimeReset, used to reset the timers RpWhile.
       "
    REFERENCE
       "802.1Qau clause 32.11.2"
    DEFVAL { 15 }
    ::= { ieee8021CnRpGroupEntry 6 }

ieee8021CnRpgByteReset OBJECT-TYPE
    SYNTAX      Unsigned32
    UNITS       "octets"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
       "This object controls the value for all of the state machine
        variables, rpgByteReset, used to reset the counters
        rpByteCount.
       "
    REFERENCE
       "802.1Qau clause 32.11.3"
    DEFVAL { 150000 }
    ::= { ieee8021CnRpGroupEntry 7 }

ieee8021CnRpgThreshold OBJECT-TYPE
    SYNTAX      Unsigned32
```

```
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object controls the number of times rpByteStage or
            rpTimeStage can count before the
            RP rate control state machine advances states.
           "
        REFERENCE
           "802.1Qau clause 32.11.4"
        DEFVAL { 5 }
        ::= { ieee8021CnRpGroupEntry 8 }

    ieee8021CnRpgMaxRate OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "Mbit/s"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object controls the maximum rate, in multiples of 1 Mbit/s,
            at which an RP can transmit. Default value is the speed of the
            port. A system SHALL support a minimim  value for this object
            that is no larger than 5 Mbits/s (object value 5). This rate
            includes all bits consequent to transmitting the frame on the
            LAN, including preamble, inter-frame gap, etc.
           "
        REFERENCE
           "802.1Qau clause 32.11.5"
        ::= { ieee8021CnRpGroupEntry 9 }

    ieee8021CnRpgAiRate OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "Mbit/s"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object controls the transmission rate increment in the
            RPR_ACTIVE_INCREASE state (rpgAiRate) in multiples of 1 Mbit/s.
            This rate includes all bits consequent to transmitting the
            frame on the LAN, including preamble, inter-frame gap, etc.
           "
        REFERENCE
           "802.1Qau clause 32.11.6"
        DEFVAL { 5 }
        ::= { ieee8021CnRpGroupEntry 10 }

    ieee8021CnRpgHaiRate OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "Mbit/s"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object controls the transmission rate increment in the
            RPR_HYPER_INCREASE state (rpgHaiRate) in multiples of 1 Mbit/s.
            This rate includes all bits consequent to transmitting the
```

```
            frame on the LAN, including preamble, inter-frame gap, etc.
           "
        REFERENCE
           "802.1Qau clause 32.11.7"
        DEFVAL { 50 }
        ::= { ieee8021CnRpGroupEntry 11 }

    ieee8021CnRpgGd OBJECT-TYPE
        SYNTAX      Integer32
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object controls the number of bits that the value of the
            Quantized Feedback field received in a CNM PDU is shifted to
            the right to decrease rpTargetRate. rpgGd is thus 2 to the
            negative power of this object, e.g., 7 means rpgGd = 1/128.
           "
        REFERENCE
           "802.1Qau clause 32.11.8"
        DEFVAL { 7 }
        ::= { ieee8021CnRpGroupEntry 12 }

    ieee8021CnRpgMinDecFac OBJECT-TYPE
        SYNTAX      Unsigned32 (1..100)
        UNITS       "percent"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object controls the minimum factor by which the current
            RP transmit rate rpCurrentRate can be changed by reception
            of a Congestion Notification Message.  Its integer value
            represents a percentage, from 1% to 100%.
           "
        REFERENCE
           "802.1Qau clause 32.11.9"
        DEFVAL { 50 }
        ::= { ieee8021CnRpGroupEntry 13 }

    ieee8021CnRpgMinRate OBJECT-TYPE
        SYNTAX      Unsigned32
        UNITS       "Mbit/s"
        MAX-ACCESS  read-write
        STATUS      current
        DESCRIPTION
           "This object controls the minimum transmission rate (rpgMinRate)
            in multiples of 1 Mbit/s.  A system SHALL support a value for
            this object that is no larger than 5 Mbit/s per second.
            This rate includes all bits consequent to transmitting the
            frame on the LAN, including preamble, inter-frame gap, etc.
           "
        REFERENCE
           "802.1Qau clause 32.11.10"
        DEFVAL { 5 }
        ::= { ieee8021CnRpGroupEntry 14 }
```

```
-- ****************************************************************
-- IEEE 802.1Qau MIB Module - Conformance Information
-- ****************************************************************

ieee8021CnCompliances OBJECT IDENTIFIER ::= { ieee8021CnConformance 1 }
ieee8021CnGroups      OBJECT IDENTIFIER ::= { ieee8021CnConformance 2 }

-- ****************************************************************
-- Units of conformance
-- ****************************************************************

ieee8021CnGlobalReqdGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnGlobalMasterEnable,
      ieee8021CnComPriLldpInstanceChoice,
      ieee8021CnComPriLldpInstanceSelector
    }
    STATUS      current
    DESCRIPTION
       "Objects in the global required group."
    ::= { ieee8021CnGroups 1 }

ieee8021CnCpGlobalGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnGlobalCnmTransmitPriority,
      ieee8021CnGlobalDiscardedFrames
    }
    STATUS      current
    DESCRIPTION
       "Objects in the Congestion Point global group."
    ::= { ieee8021CnGroups 2 }

ieee8021CnCpidTranslateGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnCpidToIfComponentId,
      ieee8021CnCpidToIfIfIndex,
      ieee8021CnCpidToIfCpIndex
    }
    STATUS      current
    DESCRIPTION
       "Objects in the CPID translate group."
    ::= { ieee8021CnGroups 3 }

ieee8021CnEplGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnEpIfIndex
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Errored Ports Table group."
    ::= { ieee8021CnGroups 4 }
```

```
ieee8021CnComPriGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnComPriDefModeChoice,
      ieee8021CnComPriAdminDefenseMode,
      ieee8021CnComPriCreation,
      ieee8021CnComPriRowStatus
    }
    STATUS      current
    DESCRIPTION
       "Objects for the global per-priority group."
    ::= { ieee8021CnGroups 5 }

ieee8021CnCpPriGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnComPriAlternatePriority,
      ieee8021CnComPriAutoAltPri
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Congestion Point per-priority group."
    ::= { ieee8021CnGroups 6 }

ieee8021CnGlobalPriPortGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnPortPriDefModeChoice,
      ieee8021CnPortPriAdminDefenseMode,
      ieee8021CnPortPriAutoDefenseMode,
      ieee8021CnPortPriLldpInstanceChoice,
      ieee8021CnPortPriLldpInstanceSelector
    }
    STATUS      current
    DESCRIPTION
       "Objects for the global per-priority per-port group."
    ::= { ieee8021CnGroups 7 }

ieee8021CnCpPriPortGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnPortPriAlternatePriority
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Congestion Point per-priority per-port
        group.
       "
    ::= { ieee8021CnGroups 8 }

ieee8021CnCpGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnCpPriority,
      ieee8021CnCpMacAddress,
      ieee8021CnCpIdentifier,
      ieee8021CnCpQueueSizeSetPoint,
      ieee8021CnCpFeedbackWeight,
```

```
        ieee8021CnCpMinSampleBase,
        ieee8021CnCpDiscardedFrames,
        ieee8021CnCpTransmittedFrames,
        ieee8021CnCpTransmittedCnms,
        ieee8021CnCpMinHeaderOctets
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Congestion Point group."
    ::= { ieee8021CnGroups 9 }

ieee8021CnRpppGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnRpPortPriMaxRps,
      ieee8021CnRpPortPriCreatedRps,
      ieee8021CnRpPortPriCentiseconds
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Reaction Point per-Port per-priority group."
    ::= { ieee8021CnGroups 10 }

ieee8021CnRpGroup OBJECT-GROUP
    OBJECTS {
      ieee8021CnRpgEnable,
      ieee8021CnRpgTimeReset,
      ieee8021CnRpgByteReset,
      ieee8021CnRpgThreshold,
      ieee8021CnRpgMaxRate,
      ieee8021CnRpgAiRate,
      ieee8021CnRpgHaiRate,
      ieee8021CnRpgGd,
      ieee8021CnRpgMinDecFac,
      ieee8021CnRpgMinRate
    }
    STATUS      current
    DESCRIPTION
       "Objects for the Reaction Point group."
    ::= { ieee8021CnGroups 11 }


-- ****************************************************************
-- MIB Module Compliance statements
-- ****************************************************************

ieee8021CnBridgeCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION
       "The compliance statement for support by a bridge of
        the IEEE8021-CN-MIB module."

    MODULE SNMPv2-MIB -- The SNMPv2-MIB, RFC 3418
        MANDATORY-GROUPS {
            systemGroup
```

```
            }

        MODULE IF-MIB -- The interfaces MIB, RFC 2863
            MANDATORY-GROUPS {
                ifGeneralInformationGroup
            }

        MODULE
            MANDATORY-GROUPS {
                ieee8021CnGlobalReqdGroup,
                ieee8021CnCpGlobalGroup,
                ieee8021CnCpidTranslateGroup,
                ieee8021CnEplGroup,
                ieee8021CnComPriGroup,
                ieee8021CnCpPriGroup,
                ieee8021CnGlobalPriPortGroup,
                ieee8021CnCpPriPortGroup,
                ieee8021CnCpGroup
             }

        OBJECT ieee8021CnComPriRowStatus
          SYNTAX       RowStatus { active(1), notInService(2),
                                   notReady(3) }
          WRITE-SYNTAX RowStatus { notInService(2), createAndGo(4),
                                   destroy(6) }
          DESCRIPTION "Support for createAndWait is not required."

        ::= { ieee8021CnCompliances 1 }

    ieee8021CnStationCompliance MODULE-COMPLIANCE
        STATUS      current
        DESCRIPTION
           "The compliance statement for support by an end station of
            the IEEE8021-CN-MIB module."

        MODULE SNMPv2-MIB -- The SNMPv2-MIB, RFC 3418
            MANDATORY-GROUPS {
                systemGroup
            }

        MODULE IF-MIB -- The interfaces MIB, RFC 2863
            MANDATORY-GROUPS {
                ifGeneralInformationGroup
            }

        MODULE
            MANDATORY-GROUPS {
                ieee8021CnGlobalReqdGroup,
                ieee8021CnComPriGroup,
                ieee8021CnGlobalPriPortGroup,
                ieee8021CnRpppGroup,
                ieee8021CnRpGroup
             }

```

```
        GROUP ieee8021CnCpGlobalGroup
            DESCRIPTION
               "This group is optional and supports end stations that
                choose to implement Congestion Points."

        GROUP ieee8021CnCpidTranslateGroup
            DESCRIPTION
               "This group is optional and supports end stations that
                choose to implement Congestion Points."

        GROUP ieee8021CnEplGroup
            DESCRIPTION
               "This group is optional and supports end stations that
                choose to implement Congestion Points."

        GROUP ieee8021CnCpPriGroup
            DESCRIPTION
               "This group is optional and supports end stations that
                choose to implement Congestion Points."

        GROUP ieee8021CnCpPriPortGroup
            DESCRIPTION
               "This group is optional and supports end stations that
                choose to implement Congestion Points."

        GROUP ieee8021CnCpGroup
            DESCRIPTION
               "This group is optional and supports end stations that
                choose to implement Congestion Points."

        ::= { ieee8021CnCompliances 2 }
    END
```

Insert a new Clause 30 as follows.

30. Principles of congestion notification

Congestion notification comprises capabilities for detecting and mitigating queue congestion for selected classes of traffic in Virtual Bridged Local Area Networks. These capabilities can be used in networks with limited bandwidth-delay products in order to decrease the likelihood of frame loss for Congestion Controlled Flows (CCFs).

Congestion notification depends on the formation of a cooperating set of systems including VLAN-aware Bridges and end stations to achieve the reduction in frame loss. By partitioning the bridges' and end stations' resources, frames sourced or sunk by non-cooperating end stations or bridges can be carried with minimal contention with CCFs for those resources. Accordingly, congestion notification entities (Clause 31) are specified as shims that make use of and provide the ISS or EISS (6.6, 6.8) at Service Access Points (SAPs) within the network. The operation of the congestion notification protocol is described in Clause 32, and the formats of the Congestion Notification Message and Congestion Notification TLV are described in Clause 33.

This clause provides the context necessary to understand the congestion notification protocol: how congestion controlled regions of networks are identified; how CCFs are identified and separated from other frames; and how congestion notification entities in bridges and end stations operate.

This Clause introduces the concepts essential to congestion notification:

a) The problem and requirements on the solution are presented (30.1).
b) The basic principles of the Quantized Congestion Notification protocol (QCN) are presented, including descriptions of a Congestion Point (CP) that sends a Congestion Notification Message (CNM) to a Reaction Point (RP) (30.2);
c) A Congestion Controlled Flow (CCF) consists of frames, all with the same priority value, and all assigned to a single Flow queue and Reaction Point in the originating end station (30.3);
d) A Congestion Notification Priority Value (CNPV) is one of the eight priority values that has been configured exclusively for the use of CCFs in the Congestion aware systems of a Virtual Bridged LAN (30.4);
e) A Congestion Notification Tag can be transmitted with every data frame in a CCF, and is returned in every CNM (30.5);
f) A Congestion Notification Domain (CND) is constructed as a subset of bridges and end stations in a Virtual Bridged LAN, all of which are configured to handle a particular CNPV, the resources of which subset are defended by the bridges whenever the offered load exceeds the network capacity (30.6);
g) The relationship of Multicast data to congestion notification is explained (30.7); and
h) The peering relationship of RPs and CPs in Provider Bridged and Provider Backbone Bridged Networks is explained (30.8).

30.1 Congestion notification design requirements

The congestion notification protocol, congestion notification algorithm (QCN), and supporting protocols specified in this standard meet requirements for the following:

— CCF and network performance
— Limiting the implementation complexity, of both RPs and CPs
— Not over-constraining the design of Bridged Local Area Networks using CN

— Facilitating interoperability, coexistence, and deployment into existing networks, and limiting the administrative effort associated with using CN in a network.

NOTE 1—QCN is designed to operate over a wide range of conditions. This clause (30.2) necessarily uses qualitative terms such as “low probability”, “small”, “stable”, and “modest” (for brevity). The bibliography (B) provides references [B44] to studies of QCN performance for various network and traffic configurations, using both simulation and a fluid model (see 30.3 for the necessary equations, and for network sizing recommendations).

The following performance requirements are met for Congestion Controlled Flows (CCFs, 30.3):

a) There is a very low probability of frame discard due to buffer overflow at a CP,
b) The average buffer occupancy at each CP that is a Current Congestion Point (CCP) for one or more CCFs is a small fraction of the bandwidth delay product, and largely independent of the number of CCFs, thus ensuring that loss avoidance does not result in large and fluctuating transit delays.
c) The buffer occupancy at a CCP has a low probability of underflow, i.e. of becoming empty and thus failing to use available bandwidth when an RP is rate limiting a CCF as a result of CNMs received from that CCP.
d) When multiple CCFs with a single CCP share that CCP, they obtain very nearly the same share of the bandwidth. When multiple CCFs have one or more CCPs in common they each obtain a share of bandwidth resources proportional to their share of contested resources.
e) The low probability of buffer overflow and underflow at a CCP is maintained for both small and large numbers of CCFs, and when CCFs or the load offered by a given CCF changes.
f) The bandwidth provided to each CP by its attached LAN will be used, at all times, i.e. if the CP’s buffer is not empty it will transmit, given the available bandwidth. The fact that the bandwidth available to one CCF passing through a given CP can be constrained by another CCP will not reduce the bandwidth through that CP for other CCFs not sharing the CCP.

NOTE 2—The probability of frame loss in the absence of CN naturally depends on the response of higher layer protocols to increasing transit delay, and the importance of avoiding loss on their response. Highly loss sensitive protocols typically lack loss avoidance mechanisms, and no such mechanisms are assumed in the analysis of QCN.

NOTE 3—Requirements (a) through (d) can be summarized by characterizing the performance of QCN as stable, responsive, and (proportionally) fair with respect to buffer occupancy processes. Requirements (e) and (f) demand that available resources are used, and that stability not depend on having a large number of slowly changing CCFs.

The following requirements limit the complexity and hence the cost of Bridge Port (CP) support for CN:

g) The CN state retained by each Bridge Port is fixed, and determined solely by the number of congestion management transmission queues (i.e. by the number of CPs, one per CNPV, up to a maximum of 7) and the per CP requirements of the congestion notification algorithm (QCN).
h) A CP does not retain any per CCF or per RP state, nor is it required to allocate any data frame passing through the CP to a CCF or an RP. All the information necessary for a CP to address a CNM to an end station, and for that end station to identify the CCF and RP from the CNM, is contained in each frame of a CCF.
i) The buffering required, at each CP, to avoid frame loss should be a small fraction of the network’s bandwidth delay product.

The following requirements limit the complexity and hence the cost of end station (RP) support for CN:

j) The CN state retained by each end station for each congestion managed transmission queue, i.e. for each RP, is fixed.
k) The end station state requirements of QCN are determined solely by the number of RPs, and are independent of the number of CCFs and the number of CPs in the network.

l) The number of CCFs, the identification of each CCF, the allocation of transmitted frames to CCFs, and the allocation of CCFs to RPs is determined by each end station independently.
m) The end station is not required to transmit any additional frames, either as a consequence of transmitting or of receiving a frame for a CCF,
n) The CN-TAG for each CCF is independent of changes in RP state information.
o) The CN protocol and QCN do not use any higher-layer protocol information carried in frames.
p) QCN calculations are simple and can be performed rapidly: i.e. the number of operations required to process each event is fixed, and all multiplications and divisions can be reduced to the use of factors of 2 or the use of a fixed and small lookup table. An RP does not attempt to calculate control loop gains and delays and other, potentially rapidly changing CCF dependent variables.

NOTE 4—A CP or an RP can retain additional state for the purposes of management, but this state is not required by the CN protocol or QCN.

NOTE 5—An end station can use higher layer protocol information to decide whether a frame is to be allocated to a CCF, and to select a CCF.

The following requirements ensure that CN does not over-constrain the design of the network, or limit the use of protocols that determine the active topology of the network:

q) RPs and CPs are unaware of the path taken by frames through the network, require no signalling from the network of changes in path, and take no exceptional action when paths change.
r) An RP does not assume that all frames that it transmits, or any identifiable subset of those frames such as those identified by the end station as belonging to a single CCF or transmitted to the same destination, are constrained to follow the same path through the network. An RP can transmit frames to many different destinations, and transmission to a given destination can be multi-path.

The following requirements support interoperability, facilitate deployment, and limit the administrative effort required to use CN:

s) The boundaries of each Congestion Notification Domain(CND) are determined automatically.
t) The destination of a CCF can lie outside the transmitting end station's CND, and the destination end station can be unaware of the use of CN within the CND. Incremental deployment of CN is therefore possible.
u) The feedback provided in each CNM provides the receiving RP with the information needed to act on that CNM, and does not require the RP to know or interpret information about the sending CP.
v) Provider Backbone Bridged Networks can be interposed into a CND, in order to reduce the number of MAC addresses leaned by bridges in environments with large numbers of (perhaps virtual) end stations.

These requirements have a number of additional consequences for the design of QCN, including the following:

w) Additional mechanisms to determine round trip time are not required, as the algorithm is robust over its targeted range of round trip time.
x) QCN cannot regulate frame transmission by acknowledgements, as does TCP/IP.

30.2 Quantized Congestion Notification protocol

This subclause provides an overview of the baseline simulation of the QCN algorithm [B44] used to develop this Standard. This introduction provides no normative text. It will focus on the key features of the QCN algorithm, omitting the normative details given in the rest of this Standard.

The QCN algorithm is composed of two parts:

a) **Congestion Point (CP) Algorithm:** this is the mechanism by which a congested bridge or end station buffer samples outgoing frames and generates a feedback message (Congestion Notification Message or CNM, 33.3, in this standard) addressed to the source of the sampled frame. The feedback message contains information about the extent of congestion at the CP.
b) **Reaction Point (RP) Algorithm:** this is the mechanism by which a Rate Limiter (RL) associated with a source decreases its sending rate based on feedback received from the CP, and increases its rate *unilaterally* (without further feedback) to recover lost bandwidth and probe for extra available bandwidth.

30.2.1 The CP Algorithm

A bridge containing a CP is modeled as an ideal output-queued bridge. The CP buffer is shown in Figure 30-1. The goal of the CP is to maintain the buffer occupancy at a desired operating point, Q_{eq}.[1] The CP computes a congestion measure F_b (defined below) and, with a probability depending on the severity of congestion, selects a frame from the incoming stream and sends the value of F_b in a feedback message to the source of the selected frame.

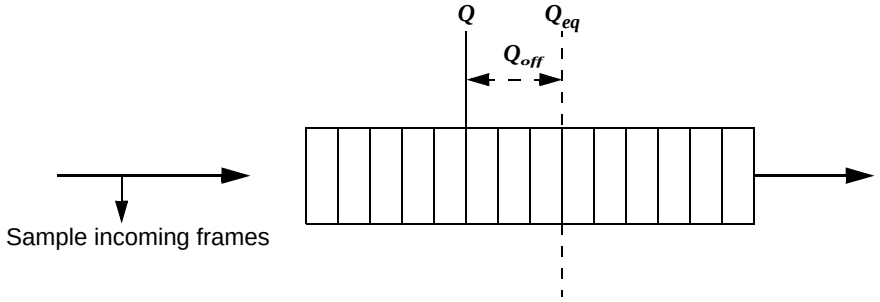

Figure 30-1—Congestion detection in QCN CP

Let Q denote the instantaneous queue size and Q_{old} denote the queue size when the last feedback message was generated. Let $Q_{off} = Q - Q_{eq}$ and $Q_\delta = Q - Q_{old}$.

Then F_b is given by the formula

$$F_b = -(Q_{off} + wQ_\delta) \quad (1)$$

where w is a non-negative constant, taken to be 2 for the baseline simulation. The value of F_b is quantized to 6 bits before transmission, based on Q_{eq} and w.

The interpretation is that F_b captures a combination of queue size excess (Q_{off}) and rate excess (Q_δ). Indeed, $Q_\delta = Q - Q_{old}$ is the *derivative* of the queue size and equals input rate less output rate. Thus, when F_b is negative, the buffer is oversubscribed. When $F_b < 0$, Figure 30-2 shows the probability with which a congestion message is reflected back to the source as a function of $|F_b|$. The feedback message contains the value of F_b, quantized to 6 bits. When $F_b \geq 0$, there is no congestion and no feedback messages are sent.

30.2.2 Basic Reaction Point Algorithm

The Reaction Point (RP) is not given positive rate-increase signals by the network. Also, there are no acks at Layer 2 to trigger rate increases, Therefore, the RP requires a local mechanism to determine when, and by

1. In simulation, setting Q_{eq} to 20% of the physical buffer size (150,000 octets) produced good results.

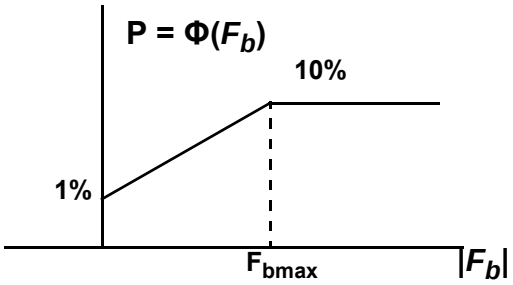

Figure 30-2—Sampling (reflection) probability in QCN CP as a function of $|F_b|$

how much, the send rate is increased. Before proceeding to explain the RP algorithm, we will need the following terminology:

- **Current Rate (*CR*):** The transmission rate of the Rate Limiter (RL) at any time.
- **Target Rate (*TR*):** The sending rate of the RL *just before* the arrival of the last feedback message, and the new goal for the Current Rate.
- **Byte Counter:** A counter at the RP for counting the number of bytes transmitted by the RL. It triggers rate increases by the RL. See 30.2.2.2.
- **Timer:** A clock at the RP which is also used for triggering rate increases at the RL. The main purpose of the timer is to allow the RL to rapidly increase the rate when its sending rate is very low and bandwidth becomes available. See 30.2.3.

We now explain the RP algorithm assuming that only the byte counter is available. Later, we will briefly explain how the timer is integrated into the RP algorithm. Figure 30-3 shows the basic RP behavior.

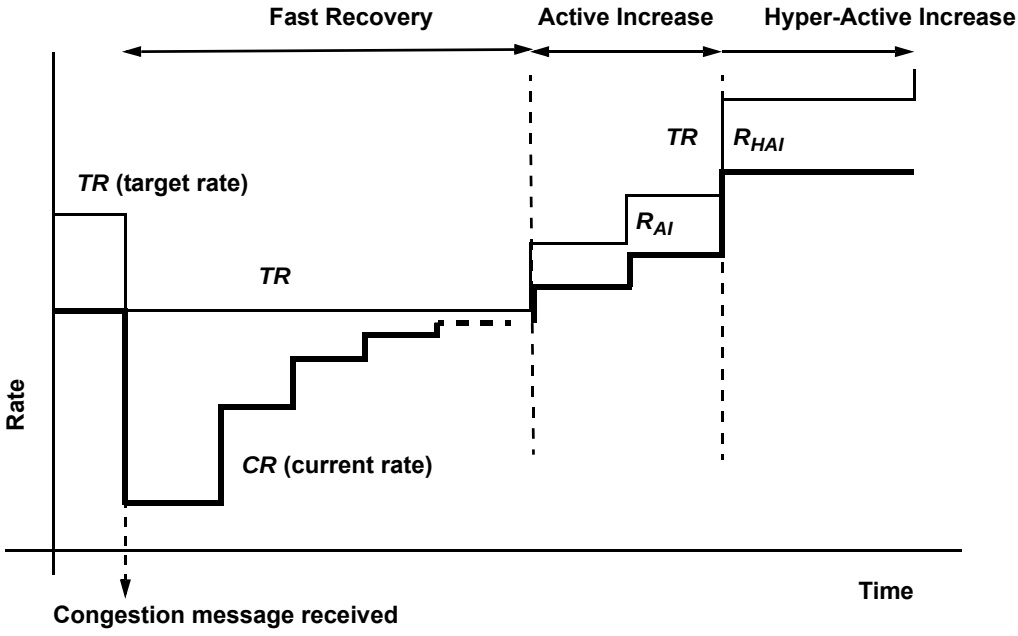

Figure 30-3—QCN RP operation

30.2.2.1 Rate decreases

Rate decreases occur only when a feedback message is received, in which case *CR* and *TR* are updated as follows:

$$TR \leftarrow CR \quad (2)$$
$$CR \leftarrow CR(1 - G_d|F_b|) \quad (3)$$

where the constant G_d is chosen so that $G_d|F_{bmax}| = ½$, i.e., the sending rate can decrease by at most 50%.

30.2.2.2 Rate increases

Rate increases occur in two phases: Fast Recovery and Active Increase.

Fast Recovery (FR). The byte counter is reset every time a rate decrease is applied and the RP enters the FR state. FR consists of 5 cycles, each cycle equal to 150 KBytes of data transmission by the RL. The RP counts the cycles of the byte counter. At the end of each cycle, *TR* remains unchanged while *CR* is updated as follows:

$$CR \leftarrow ½(CR + TR) \quad (4)$$

The cycle duration of 150 KBytes is chosen to correspond to the transmission of 100 frames, each 1500 Bytes long. The idea is that when the RL has transmitted 100 frames and, given that the minimum sampling probability at the CP is 1%, if it hasn't received a feedback message then it may infer that the CP is uncongested. Therefore it increases its rate as above, recovering some of the bandwidth it lost at the previous rate decrease episode. Thus, the goal of the RP in FR is to *rapidly recover* the rate it lost at the last rate decrease episode.[2]

Active Increase (AI). After 5 cycles of FR have completed, the RP enters the Active Increase (AI) phase where it probes for extra bandwidth on the path. During AI, the byte counter counts out cycles of 50 frames (this can be set to 100 frames for a less frequent probing). At the end of each cycle, the RL updates *TR* and *CR* as follows:

$$TR \leftarrow TR + R_{AI} \quad (5)$$
$$CR \leftarrow ½(CR + TR) \quad (6)$$

where R_{AI} is a constant chosen to be 5 Mbps in the baseline simulation.

30.2.3 RP algorithm with timer

Since rate increases using the Byte Counter occur at a frequency proportional to the current sending rate of the RL, when *CR* is very small, the duration of Byte Counter cycles when measured in seconds can become unacceptably large. Since the speed of bandwidth recovery (or responsiveness) is a key performance metric, a Timer was included in QCN.

The Timer functions similarly as the Byte Counter: it is reset when a feedback message arrives, enters FR and counts out 5 cycles of T ms duration (T is 10 ms long in the baseline), and then enter AI where each cycle is T/2 ms long.

The Byte Counter and Timer jointly determine rate increases at the RL as shown in Figure 30-4. After a feedback message is received, they each operate independently and execute their respective cycles of FR and AI. Together, they determine the state of the RL and its rate changes as follows:

a) The RL is in FR if *both* the Byte Counter and the Timer are in FR. In this case, when either the Byte Counter or the Timer completes a cycle, CR is updated according to (4).
b) The RL is in AI if *only one of* the Byte Counter and the Timer is in AI. In this case, when either completes a cycle, *TR* and *CR* are updated according (5) and (6).

2. The BIC-TCP algorithm [B45] has a similar Fast Recovery mechanism.

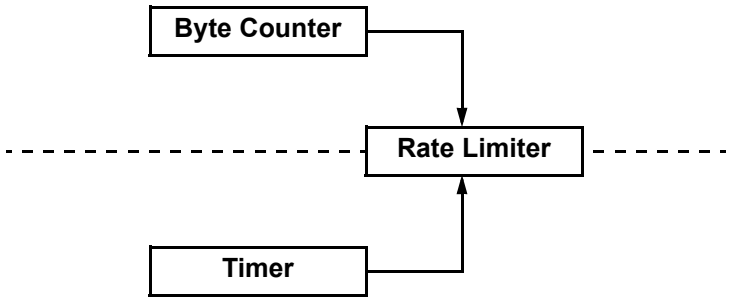

Figure 30-4—Byte Counter and Timer interaction with Rate Limiter

c) The RL is in HAI (for Hyper-Active Increase) if *both* the Byte Counter and the timer are in AI. In this case, the *i*th time that a cycle for either the Byte Counter or the Timer is completed, *TR* and *CR* are updated as:

$$TR \leftarrow TR + iR_{HAI} \quad (7)$$

$$CR \leftarrow \tfrac{1}{2}(CR + TR) \quad (8)$$

where R_{HAI} is constant set to 50 Mbps in the baseline simulation (Figure 30-3). So the increments to *TR* in HAI occur in multiples of 50 Mbps.

Thus, the Byte Counter and Timer should be viewed as providing the RL opportunities to increase the rate. Their state determines the state of, and hence the amount of rate increase at, the RL.

It is very important to note that the RL goes to HAI only after at least 500 frames have been sent and 50 msecs have passed since the last congestion feedback message was received. This doubly ensures that aggressive rate increases occur only after the RL provides the network adequate opportunity (in frames sent for possible sampling) for sending rate decrease signals should there be congestion. This is vital to ensure the stability of the algorithm, and while optimizations can be performed to improve its responsiveness, in the interests of stability and simplicity, the baseline simulation [B44] took no further optimization steps.

30.3 Congestion Controlled Flow

A Congestion Controlled Flow (CCF) consists of all of the frames passing through a single Flow queue (31.2.2.1) in an originating end station. Examples of CCFs include, but are not limited to:

a) The frames carrying data for a single User Datagram Protocol (UDP, IETF RFC768, STD0006) connection.
b) All of the frames generated by a particular process running in an end station.
c) All of the frames being transmitted by an end station that produce the same value when a hash computation is performed on the source and destination IP addresses, UDP vs. TCP selection, and source and destination UDP/TCP port numbers of the IP packets carried by those frames.
d) All of the frames passing through an RP that have the same destination_address parameter.
e) All of the frames being transmitted by an end station.
f) All of the frames passing through a Bridge Port sharing identical source_address, destination_address, and priority parameters.
g) All of the frames leaving a router that contain IP packets, and have identical values for their source and destination IP addresses, UDP vs. TCP selection, and source and destination UDP/TCP port numbers.
h) All of the frames with a certain value for the priority parameter.

All of the frames in a given CCF are sent with the same priority value, which is a CNPV (30.4).

Not all frames transmitted by an end station with an RP need be CCFs. Frames not belonging to CCFs bypass the Flow queues and RPs.

30.4 Congestion Notification Priority Value

A Congestion Notification Priority Value (CNPV) consists of one value of the priority parameter such that all of the bridges' and end stations' ports in a Congestion Notification Domain are configured to assign frames at that value to the same CP and/or an RP. Different CNPVs correspond to classes of applications, or even single applications, such as interprocess communications or disk storage data, that have different requirements for such network resources as latency or bandwidth. Since there are eight values for the priority parameter, a single port in a Virtual Bridged Network can support as many as seven CNPV values; at least one value is required for best-effort traffic.

30.5 Congestion Notification Tag

An end station may add a Congestion Notification Tag (CN-TAG) to every frame it transmits from a CCF. The CN-TAG contains a Flow Identifier field. The destination_address, Flow Identifier, and a portion of the frame that triggered the transmission of the CNM are the means provided by this Standard by which a station can determine to which RP a CNM applies. How the station makes that determination is beyond the scope of this Standard. The reasons for adding a CN-TAG include:

a) An end station can identify a particular flow within a CCF that triggered the CNM;
b) It can be simpler for the end station to use a Flow Identifier, rather than to parse the returned mac_service_data_unit of the original triggering frame, in order to identify either the RP or the particular flow; and
c) Depending on the application, e.g. if fragmented IP datagrams are transmitted, it can be impossible to determine either the RP or the particular flow based solely on a returned mac_service_data_unit.

An end station that has only a single RP per CNPV, or that is able to identify the RP on the basis of the returned portion of the triggering frame, can omit transmitting the CN-TAG. The CP always returns a CN-TAG in a CNM with the triggering frame's Flow Identifier, or with a 0 Flow Identifier, if the triggering frame has no CN-TAG. The fixed format for the CNM minimizes the effort required of the CP to generate the CNM.

30.6 Congestion Notification Domain

In order for congestion notification to successfully control congestion in a Virtual Bridged Network, the managed objects controlling the congestion notification state machines in the bridges and end stations in that network have to be configured with values that are appropriate to the characteristics of the CCFs generated by the applications that expect congestion controlled services. For example:

a) If frames that were not originated from an RP can enter a CP experiencing congestion, then the CNMs generated by that CP upon receipt of those frames cannot correct the problem, and congestion notification cannot operate correctly.
b) Congestion notification cannot operate correctly if a CP's configuration is inappropriate for the CCFs passing through it, or if priority values are regenerated in a manner that moves frames in and out of CNPVs.
c) Frames transmitted from an end station with a CN-TAG cannot be understood by an end station that is not congestion aware.

Congestion aware bridges therefore construct a Congestion Notification Domain (CND), within which a particular CNPV is supported. A CND is a connected subset of the bridges and end stations in a Virtual Bridged LAN that are appropriately configured to serve a particular CNPV. This standard does not assume that every bridge in a Virtual Bridged Network will be capable of congestion control, does not assume that every capable bridge will be so configured, and does not assume that the subset of bridges in a Virtual Bridged Network forming a CND that serves one CNPV will be the same subset that form a CND serving another CNPV.

CNDs can be created by configuring the bridges and end stations in a network, or they can be created automatically, using an additional Type Length Value element, the Congestion Notification TLV (33.5), that this standard adds to the IEEE Std. 802-2004 Link Layer Discovery Protocol. Either way, every Bridge Port knows, for each CNPV, whether its neighbor(s) are or are not all configured with a matching CNPV.

NOTE—If CND boundaries are configured individually, rather than through LLDP, and there is an error in the configuration, there is a risk that frames not originating from RPs will pass through a CP, or that RP-originated frames will pass through a congested Port that has no CP. Either case can result in congestion that cannot be alleviated, with consequent excessive discarding of RP-originated data frames. In addition, such misconfiguration can result in a congestion aware end station being unable to communicate with a congestion unaware end station, because the latter is receiving CN-TAGs that it does not understand.

Making each CND/CNPV independent, rather than making a single determination of neighbor capabilities based on whether or not all CNPVs are configured the same for all Ports attached to the LAN, allows a CND to be added to or removed from a network already using another CND, without disrupting the operation of the first CND.

30.7 Multicast data

Although control of multicast streams is not an object of this Standard, and no simulations or experiments involving multicast data were utilized in the writing of this Standard, the transmission of multicast data through a CCF is not prohibited. Some applications primarily send unicast data, but send occasional multicast frames for purposes such as discovery. Transmission of multicast data has been allowed in order to allow well-designed applications using a very small percentage of multicast traffic to operate on a CNPV.

Multicast streams are often sent to multiple destinations along multiple paths through the network, and a unicast stream normally travels along a single path. Therefore, a multicast stream is more likely to encounter any congested ports that exist in a network than is a unicast stream. The consequences of sending multicast streams on a CCF include:

a) The RP could limit the CCF to the rate of the slowest (most congested) path taken by the multicast
b) The chances of unintended fate sharing are higher in a CCF that included multicast streams. That is, a multicast stream is more likely to unnecessarily slow down a unicast stream sharing the same CCF than one unicast stream is likely to slow down another.

30.8 Congestion notification and additional tags

The transmission of a CNM from a CP to an RP makes the CP and RP peer entities in the sense of 6.1. An example of this peering relationship for a VLAN Bridged Network is illustrated in Figure 30-5, with the CP–RP relationship shown by a heavy dashed line.

Provider Backbone Bridges (Clause 25) could be used in a data center environment, in order to reduce the number of MAC addresses that have to be learned in the core of the network. The I-component (5.7) of a Provider Backbone Bridge adds and removes I-TAGs (9.8) on data frames. However, as illustrated in

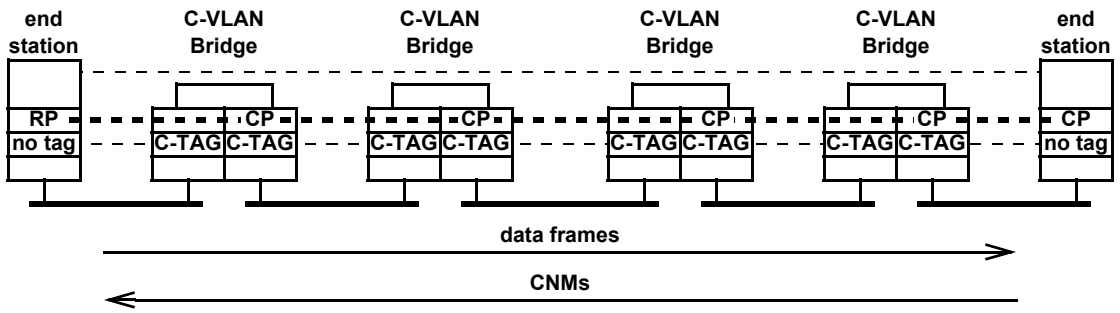

Figure 30-5—CP–RP peering in VLAN Bridged Network

Figure 30-6[3], adding an I-TAG to a data frame by an I-component impairs the ability of a CP in the core of that network to return a CNM that will be meaningful to the originating end station. The following two difficulties follow from the fact that a CP in a B-VLAN Bridge is not a peer of the RP in an end station:

a) The Backbone Core Bridge CP cannot parse the I-TAG and C-TAG, which is necessary in order to find the RP's MAC address and the data frame's CN-TAG, in order to build an I-TAG-encapsulated, CNM addressed to the RP; and
b) There is no MAC address available to the Backbone Core Bridge CP that would be a valid source_address in the CNM when received by the RP.

In order to provide the benefits of PBBNs or other hierarchical bridging technologies to the data center environment, an interworking function (32.16) is defined to allow core CPs to peer with end stations.

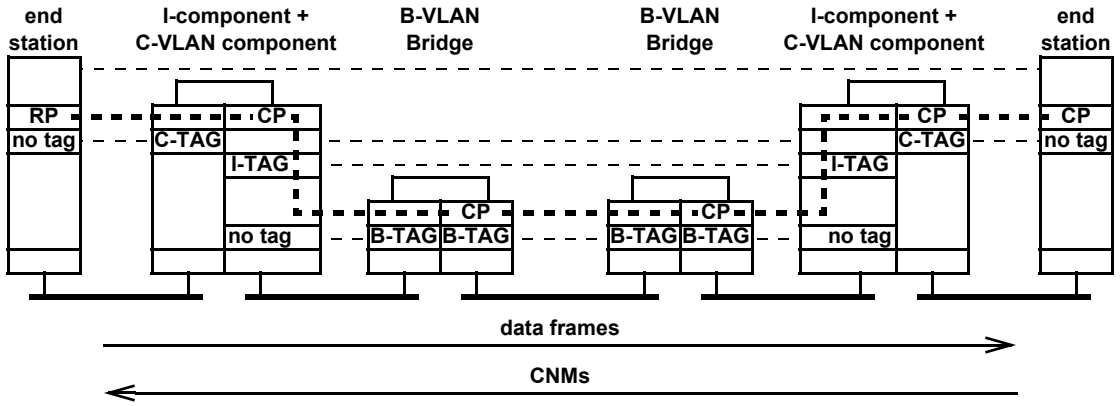

Figure 30-6—CP–RP peering in Provider Backbone Bridged Network

3. In the configuration chosen for illustration for the Data Center environment in Figure 30-6, each S-VLAN component of an I-components is, in effect, a Provider Bridge (5.10). That is, each Customer Network Port in an I-component is virtualized and multiplied, and connected to a corresponding virtual Provider Edge Port of a C-VLAN component offering a single Customer Edge Port facing the end station. This offers a C-TAG interface to the end station, without incurring the burden of an additional layer of S-TAGs.

Insert a new Clause 31 as follows.

31. Congestion notification entity operation

This clause specifies:

a) The architecture of the Congestion Point (CP) in the Forwarding Process in a congestion aware bridge component or end station (31.1); and
b) The architecture of a CP and a Reaction Point (RP) in a congestion aware end station (31.2).

NOTE—Clause 30 introduces the principles of congestion notification operation and the network architectural concepts that support it. Clause 32 specifies the protocols operated by entities defined in this Clause. Clause 33 specifies the encoding of the PDUs used by congestion notification.

The models of operation in this clause provide a basis for specifying the externally observable behavior of congestion notification, and are not intended to place additional constraints on implementations; these can adopt any internal model of operation compatible with the externally observable behavior specified. Conformance of equipment to this standard is purely in respect of observable protocol.

31.1 Congestion aware Bridge Forwarding Process

Figure 8-9 illustrates the details of the Bridge Forwarding Process. Figure 22-1 shows a different view of those details, rearranged so that those entities of the Forwarding Process that are concerned only with a single Bridge Port can be put in the proper relationship to entities such as Connectivity Fault Management shims. Figure 31-1 shows only the queuing functions of Figure 22-1, as modified for a congestion aware bridge. Two new entities are added for the congestion aware bridge: the Congestion Point (31.1.1) and the Congestion Point ingress multiplexer (31.1.2).

31.1.1 Congestion Point

As described in 32.8 and 32.9, based on the state of its internal variables, and a frame given the CP by the Queuing frames entity (8.6.6) in an EM_UNITDATA.request (parameters) (32.9.3), a CP either inserts the frame into its attached queue or discards the frame, and can generate a Congestion Notification Message (CNM), based on the frame. Discarded frames are counted in the variable cpDiscardedFrames (32.8.12) Frames successfully queued are counted in the variable cpTransmittedFrames (32.8.13). Any CNM generated is passed to the Congestion Point ingress multiplexer (31.1.2, Figure 31-1) or Flow multiplexer (31.2.4, Figure 31-2) for transmission back to the RP that sourced the frame.

Congestion notification does not operate correctly if frames not belonging to a Congestion Controlled Flow (CCF) are allowed to pass through a CP. The Congestion Notification Domain operations (32.1) help to ensure that the only frames passing through a CP belong to CCFs.

The CP shall remove the CN-TAG from every frame passing through it with a priority value (a CNPV) for which the port on which the CP resides is acting in cptEdge or cptInterior CND defense mode.

The functions of Figure 22-1 could be implemented in different ways than that illustrated, without altering the externally observed behavior of the conformant equipment. For example, the CP could pass a copy of the output frame triggering a CNM to a function in the higher layer interfaces of a bridge, which in turn, could generate the CNM. In order to provide a management interface that is useful over a wide range of implementations, however the following provisions are made:

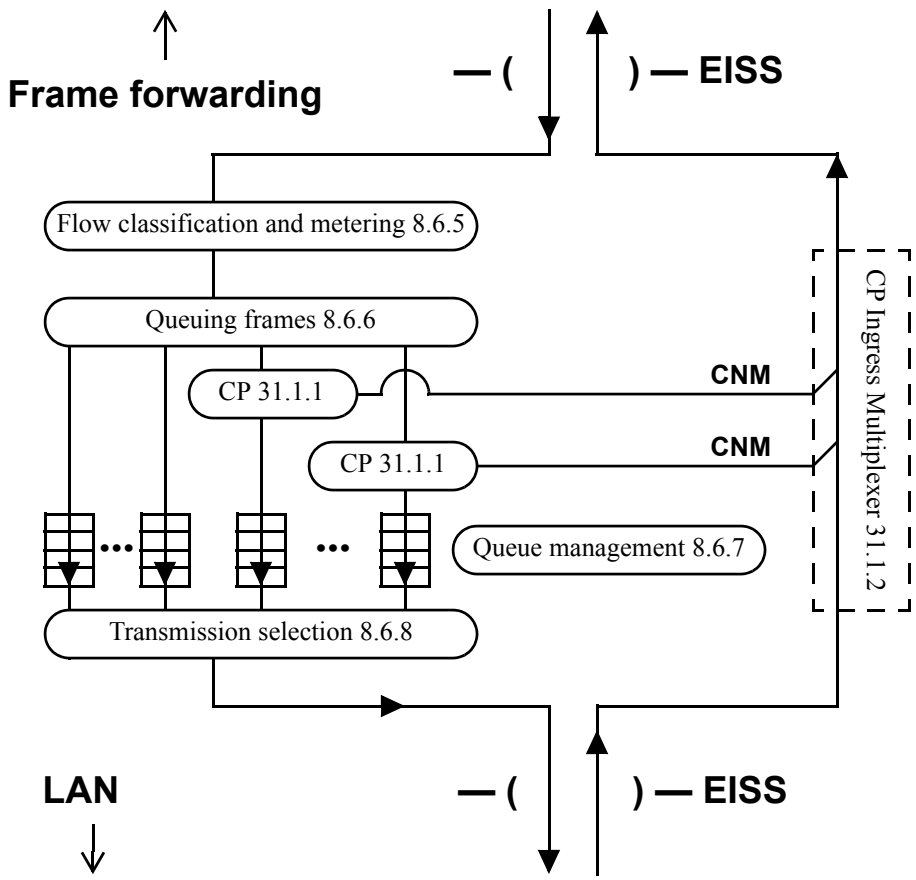

Figure 31-1—Congestion Points and congestion aware queues in a bridge

a) CNMs generated on a Bridge Port are not counted in the Frames Received (12.6.1.1.3:a) or Octets Received (12.6.1.1.3:b) counters; and
b) CNMs generated on one Bridge Port are counted, as are ordinary data frames, on the Bridge Ports (if any) on which they are output.

If a bridge supports the dot1agCfmVlanTable or the ieee8021CfmVlanTable, its CPs transmit each CNM to the Primary VID associated with the vlan_identifier of the frame triggering the CNM's transmission.

31.1.2 Congestion Point ingress multiplexer

Inserts the CNMs generated by the Congestion Points (31.1.1) among the frames received from the LAN.

31.2 Congestion aware end station functions

Figure 31-2 illustrates the architecture of the queue functions of a congestion aware end station. These functions offer a service to higher layers through a single instance of the ISS or EISS, and utilize an instance of the ISS or EISS to connect to the lower layers. The Output flow segregation function is presumed to exercise control over the higher layers' ability to present frames for transmission through the ISS or the EISS, and the Reception selection function to control the flow of frames to the higher layers, by means of additional parameters local to this instance of the ISS or the EISS and/or locally significant LMI parameters (6.1.3), neither of which are specified by this standard.

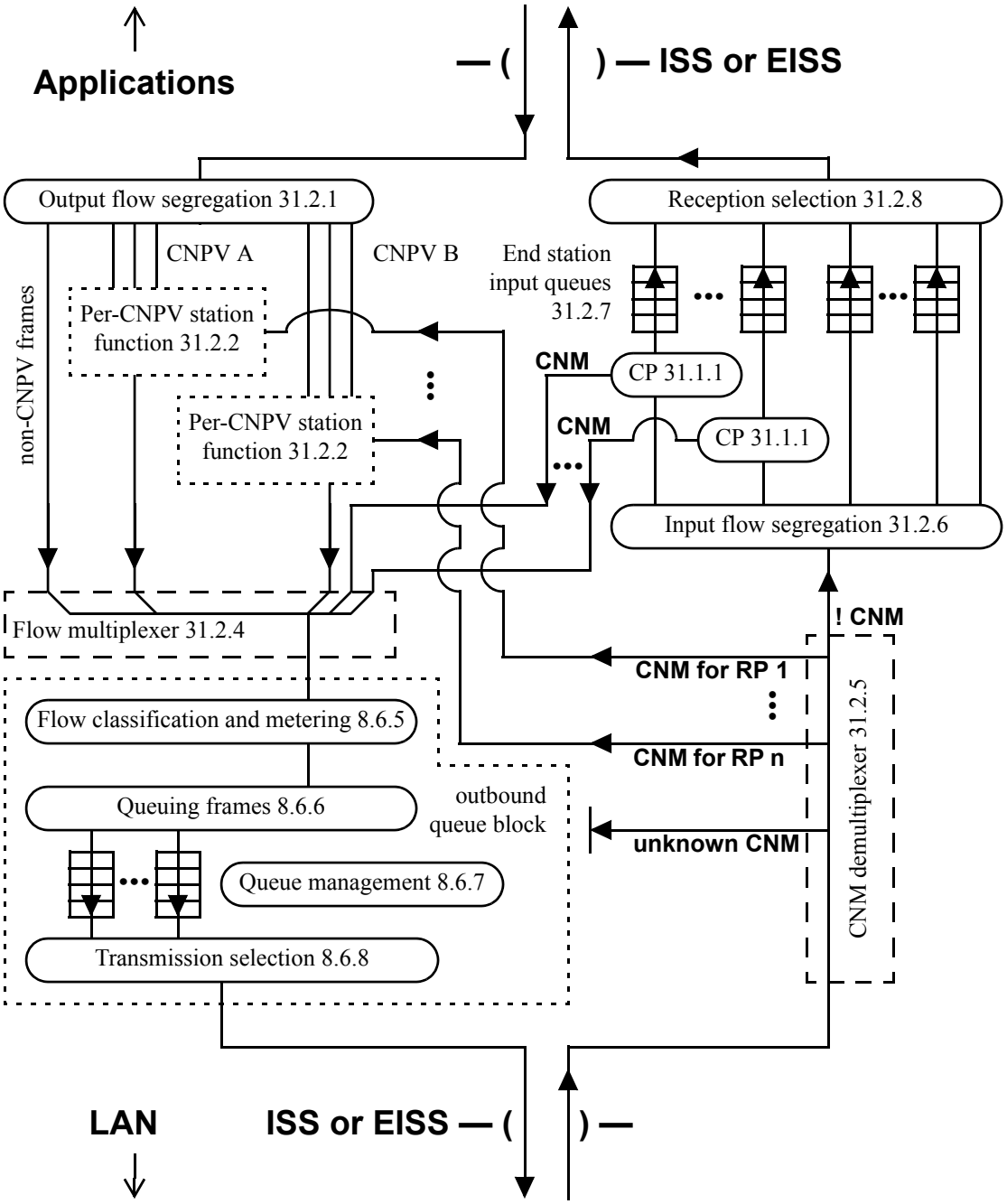

Figure 31-2—Congestion aware queue functions in an end station

In Figure 31-2, the outbound queue block, in the lower left of Figure 31-2, is exactly the same as the output queues in a Bridge Port that is not congestion aware, as illustrated in Figure 22-1.

The remaining entities illustrated in Figure 31-2 are those that are peculiar to a congestion aware end station, and are further described in the following subclauses.

31.2.1 Output flow segregation

For each frame received from the upper layers for transmission, the Output flow segregation entity:

a) Assigns the frame a priority parameter based on the priority presented from the upper layer and/or upon other criteria, unspecified by this standard;
b) Determines, using priority value and the Flow Select Database (31.2.3), to which Flow queue, if any, the frame is to be assigned;
c) If the frame is not assigned to any Flow queue (the frame's priority is not a CNPV), passes the frame directly to the Flow multiplexer entity (31.2.4);
d) If the frame is assigned to a particular Flow queue (the frame's priority is a CNPV), enqueues the frame in the selected Flow queue; and
e) For frames that are assigned to a Flow queue, depending on the state of the Flow queue to which the frame is assigned, and through means unspecified by this standard, can influence the higher layers' presentation of further frames to the Output flow segregation entity.

The Output flow segregation entity never discards a frame. Every frame presented to it is either passed to the Flow multiplexer entity (31.2.4) or stored in a Flow queue, as described, above. This standard assumes that the Output flow segregation entity can apply means, unspecified by this standard, to prevent the upper layers from offering a frame that should be stored in a Flow queue, but for which the Flow queue has insufficient resources.

The default configuration for an end station shall be to have a single Flow queue per CNPV. An end station may support additional Flow queues, or support more than one CNPV with a single Flow queue.

NOTE—Requirements for a default configurations do not imply that explicit action via SNMP is required to change from the default. For example, managed objects in an end station can be adjusted by an application program.

If the Output flow segregation function in an end station can change its method for assigning frames to Flow queues, it shall not do so in a manner that impairs the ability of any compliant CP implementation to throttle the data from the sourcing end station, and shall not materially increase the likelihood of misordered frame transmissions in a manner that adversely impacts the higher layers' functions.

31.2.2 Per-CNPV station function

There is one Per-CNPV station function on each Port of a congestion aware end station for each priority value that is a CNPV. All of the frames passing through a Per-CNPV station function have the same CNPV priority parameter value.

As illustrated in Figure 31-3, each Per-CNPV station function is composed of the following entities:

a) One or more Flow queues (31.2.2.1);
b) One or more Reaction Points (31.2.2.2), each associated with exactly one Flow queue, each comprising:
    1) A Reaction Point State Machine (31.2.2.3) for each Reaction Point; and
    2) A Rate Limiter (31.2.2.4) for each Reaction Point; and
c) One Flow Selection function (31.2.2.5).

31.2.2.1 Flow queue

A Flow queue is a first-in-first-out queue of frames, all for the same CNPV. An Output flow segregation entity selects the Flow queue into which a given frame is inserted, and a Reaction Point determines when a frame is removed from a Flow queue.

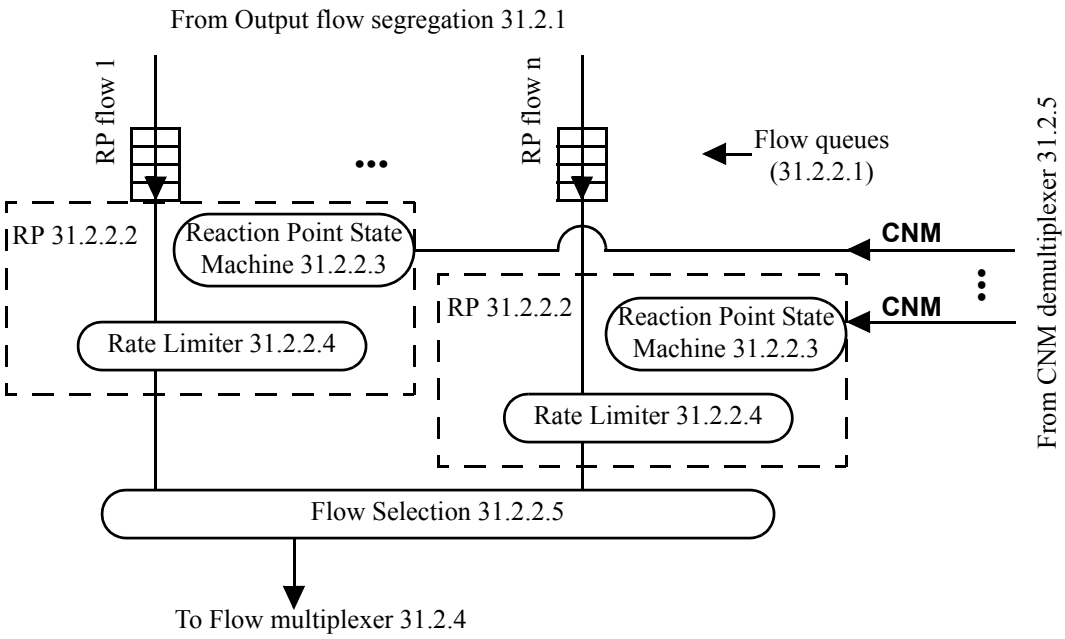

Figure 31-3—Per-CNPV station function

NOTE—The Flow queue is a notional convention used to unambiguously describe the behavior of the RP. An end station can implement the Flow queue without explicitly storing frames in a queue, based on its knowledge of the state of the higher-layer applications supplying the frames.

31.2.2.2 Reaction Point

A single Reaction Point (RP) is associated with each Flow queue. An RP is composed of a Reaction Point State Machine (31.2.2.3) and a Rate Limiter (31.2.2.4)

31.2.2.3 Reaction Point State Machine

A single Reaction Point State Machine is associated with each Flow queue. As described in detail in Clause 32, the Reaction Point State Machine consists of a timer (32.12), variables (32.10, 32.11, and 32.13), procedures (32.14), and a state machine (32.15) that determine the value of the rpLimiterRate variable (32.13.8), which, via the Rate Limiter (31.2.2.4), controls the transmission of frames from the Flow queue. The state machine and variables, in turn, are controlled by the frames input, managed objects (12.17), the Output flow segregation entity (31.2.1), and the CNMs received from the CNM demultiplexer (31.2.5)

31.2.2.4 Rate Limiter

A single Rate Limiter is associated with each Flow queue. The Rate Limiter enforces the output rate, rpLimiterRate (32.13.8), determined by the Reaction Point State Machine (31.2.2.3). The octets in the frames passed from the Rate Limiter to the Flow Selection function (31.2.2.5) are counted in rpByteCount (32.13.2). The details of the operation of the Rate Limiter are not specified by this Standard, but the measurable output rate is specified.

For a given measurement period T, we define R_T as the integral of rpLimiterRate over T, and L_T as the number of octets actually output to the LAN, including preambles, inter-frame gaps, etc., stemming from frames that pass through that RP's Rate Limiter. An end station shall ensure that:

$$L_T \leq (1.05 * R_T) + (16 \text{ maximum-sized frames}) \quad (1)$$

for the two values of T = one second and T = 1 million bits divided by the physical line rate (in bits per second), where any number of simultaneous measurements can start at any time.

NOTE 1—The rate computations and variables (equation 1, 32.13.6, 32.11.6, etc.) include all of the octets associated with a frame on the LAN, including preambles, inter-frame gaps, etc., and thus reflect actual physical data rates. However, the resource actually protected by congestion notification is octets in buffers, not octet times on wires. Per-frame overhead on the LAN is added to serve as a reasonable approximation, made at the RP, to the actual size of the frame in the Bridges' or end stations' buffers. The extra 5% margin in equation 1 is provided because this approximation is inexact.

NOTE 2—Although the simulations have been studied with a burst of one maximum-sized frame, this standard allows the end stations to send data in bursts of 16 maximum-sized frames, primarily to ease implementations. Larger bursts of data have a negative impact on the overall performance of the QCN algorithm, and can lead to lower link utilization.

31.2.2.5 Flow Selection

The Flow Selection function decides from which Flow queue in its Per-CNPV station function, if any, a frame will be offered to the Flow multiplexer. Whenever the output queue that receives frames from the Flow Selection function (via the Flow multiplexer) is full, the Flow Selection function shall not present any frame to the Flow multiplexer; this ensures that no frames in CNPVs will be discarded in the Port due to a lack of room in the output queue.

By means unspecified in this standard, the Flow Selection function selects frames to pass to the Flow multiplexer from the uncontrolled Flow queues (31.2.2.1) and Rate Limiters (31.2.2.4). Whenever a frame is passed from a Rate Limiter through the Flow Selection function to the Flow multiplexer, the Flow Selection function calls TransmitDataFrame (32.14.3) to update the Reaction Point variables (32.13). If the port's neighbor is ready to receive frames with Congestion Notification Tags for the frame's CND (defense mode is cptInteriorReady), the Flow Selection function may insert a CN-TAG (30.5) at the beginning of the frame's mac_service_data_unit, and if not (any Congestion Notification Domain defense mode other than cptInteriorReady), it shall not insert a CN-TAG. (See 33.2 for a further explanation of CN-TAG insertion.)

31.2.3 Flow Select Database

The Flow Select Database is used by the Output flow segregation (31.2.1) entity to determine to which Flow queue (31.2.2.1) a given frame is assigned.

The means by which the Flow Select Database selects a Flow queue is unspecified by this standard.

31.2.4 Flow multiplexer

The Flow multiplexer merges the frames from the Per-CNPV station functions' Flow Selection functions (31.2.2.5), the frames bypassing the Per-CNPV station functions, and the CNM frames from the Congestion Points (31.1.1), and delivers the frames to the Flow classification and metering (8.6.5) entity in the end station's output path.

31.2.5 CNM demultiplexer

The CNM demultiplexer identifies CNMs received from the LAN, uses the Flow Identifier (33.2.1) and/or Encapsulated priority (33.4.7) to determine to which Reaction Point (31.2.2.2) each CNM is intended, and directs them to the correct Reaction Points. This standard does not specify whether or how the end station can use the Flow Identifier to further identify one or more individual flows within a Flow queue and its associated RP. If a CNM is received that has no CN-TAG, or a CN-TAG whose Flow ID is unknown to the RP, the CNM is discarded. All non-CNM frames are passed to the Input flow segregation entity (31.2.6).

31.2.6 Input flow segregation

For each frame received from the lower layers for reception, the Input flow segregation entity:

a) If the frame's priority is not a CNPV, decides, in an unspecified manner, whether to pass the frame to an unspecified queue, pass it to the higher layers, or discard it;
b) If the frame's priority is a CNPV, passes the frame to a CP (31.1.1), selected by unspecified means, for processing;

NOTE 1—For example, an end station might support a single CP, or support one CP per CNPV, per VLAN, per application, or per data flow. Neither these nor any choices for CP selection are required by this standard.

The Input flow segregation entity never discards a CNPV frame. Every CNPV frame presented to it is passed to a CP, as described, above.

NOTE 2—This does not mean that a station cannot drop CNPV frames. See 31.2.7.

31.2.7 End station input queue

The number and operation of End station input queues and the method used to drain these queues (i.e., to select and transfer frames to the receiving higher layer applications) are not specified by this standard. However, the procedures performed by a CP (32.9) require that its associated End station input queue exist, and possess certain operational parameters (32.8).

In the unfortunate event that a misconfiguration or an unfortunate sequence of frame transmissions results in more CNM frames arriving at a station than it can process, an End station input queue can fill up, and discard a frame presented to it by its CP or by the Input flow segregation (31.2.6) entity.

31.2.8 Reception selection

The Reception selection entity selects frames for delivery to the upper layers. It takes the place of the Transmission selection entity (8.6.8) of a congestion aware bridge. The selection algorithm depends on interactions between the applications receiving the data, the operating system supporting those applications, and the received frames, that are not specified by this Standard. The Reception selection entity is shown in Figure 31-2 for completeness, to indicate that frames enqueued by the Queuing frames entity (8.6.6) are, in fact, removed from the queues.

If the first octets of the mac_service_data_unit of a frame selected by the Reception selection entity are a CN-TAG, Reception selection removes the CN-TAG from the mac_service_data_unit before passing the frame to the higher layers. (See 33.2 for a further explanation of CN-TAG removal.)

Insert a new Clause 32 as follows:

32. Congestion notification protocol

Congestion aware systems can participate in the congestion notification protocols specified in this clause, namely:

a) Congestion Notification Domain operations (32.1);
b) The variables controlling congestion notification at the end station or bridge component (32.2) and Port and Congestion Notification Priority Value (CNPV) levels (32.3);
c) The variables (32.4), procedures (32.5), and state machine (32.6) that control the defense of a Congestion Notification Domain;
d) The Congestion notification protocol (32.7);
e) The variables (32.8) and procedures (32.9) that define a Congestion Point (CP);
f) The variables associated with all (32.10), or a subset of (32.11), the Reaction Points (RPs) on a given Port and CNPV;
g) The timer (32.12), variables (32.13), procedures (32.14) and state machine (32.15) that define the operation of a single a Reaction Point (RP); and
h) The processing of Congestion Notification Messages by a Congestion notification and encapsulation interworking function (32.16).

NOTE—Clause 30 introduces the principles of congestion notification operation and the network architectural concepts that support it. Clause 31 breaks down the congestion notification entities into their components. Clause 33 specifies the encoding of the PDUs used by congestion notification.

32.1 Congestion Notification Domain operations

As described in 30.6, a Congestion Notification Domain (CND) is a connected subset of a Virtual Bridged Network configured to support a particular Congestion Notification Priority Value (CNPV). This standard provides a means whereby a VLAN-aware Bridge or end station can recognize the like-configured ports of a CND (32.1.2), and defend its CND against incoming frames from outside the CND.

32.1.1 Congestion Notification Domain defense

Unless every bridge along a path between two congestion aware end stations using a particular CNPV is configured for congestion notification (belongs to a CND), and unless the bridges ensure that frames not in CNPVs use different queues than the queues used by those two end stations, those end stations will not accrue the advantages that congestion notification offers. Therefore, steps must be taken at the boundaries of a CND to protect both the CND and the network outside the CND.

Every frame received on a Port passes through the ISS Support by specific MAC procedures (6.7), which can change the priority parameter of the received frame, using the Port's Priority regeneration table, Table 6-3. This table can be controlled by a managed object (12.7). In addition, in order to prevent frames not in CNPVs from entering CP-controlled queues, each Port's Priority regeneration table can be modified.

If a Port's neighbor is known to also be configured for a particular CNPV, the entry in the Port's Priority regeneration table for that CNPV is ignored, and the priority is never changed on input. The bridge component or end station connected to that Port can be trusted to use that CNPV. If the Port's neighbor is known to not be configured for a particular CNPV, the entry in the Port's Priority regeneration table for that CNPV shall be overridden to translate the CNPV to an alternate non-CNPV value. The neighboring system is not trusted, so the priority values of any frames carrying the CNPV's value are changed. Altering the priority values of received frames defends the CND from frames not originating from an RP. Similarly, if a

CNPV is configured on any Port, then on that Port, the Port's Priority regeneration table shall be overridden to prevent any other priority value from being remapped into that CNPV.

If a port has multiple neighbors, or if the neighbor on a Port is not congestion aware, then the CN-TAGs shall be removed from any frames transmitted on a CNPV. This ensures that the benefits of congestion notification can be provided as far as possible along a path from a congestion aware source to a non-congestion aware destination. Furthermore, CN-TAGs shall be removed from any frames transmitted from a CNPV towards a system that is congestion aware, but is still remapping the CNPV to a non-CNPV priority.

Thus, for each priority value, the Automatic Congestion Notification Domain recognition mode can take one of four values, whether derived from LLDP or set by managed variables:

cptDisabled: The congestion notification capability is administratively disabled for this priority value and port. This priority is not a CNPV. The priority regeneration table (6.9.3) controls the remapping of input frames on this port to or from this priority. CN-TAGs are neither added by an end station nor stripped by a bridge.

cptEdge: On this port and for this CNPV, the priority parameters of input frames are remapped to an alternate (non-CNPV) value, and no priority value is remapped to this CNPV, regardless of the priority regeneration table. CN-TAGs are not added by an end station, and are removed from frames before being output by a bridge.

NOTE—The cptEdge CND defense mode is optional for an end station. See 32.4.9.

cptInterior: On this port and for this CNPV, the priority parameters of input frames are not remapped to another value, and no priority value is remapped to this CNPV, regardless of the priority regeneration table. CN-TAGs are not added by an end station, and are removed from frames before being output by a bridge.

cptInteriorReady: On this port and for this CNPV, the priority parameters of input frames are not remapped to another value, and no priority value is remapped to this CNPV, regardless of the priority regeneration table. CN-TAGs can be added by an end station, and are not removed from frames by a bridge.

The determination of the operational state of a CNPV on a given port is determined by variables and by LLDP (32.1.2, 32.2.1, 32.3.7, 32.3.4, 32.4.1, 32.4.3).

32.1.2 Automatic Congestion Notification Domain recognition

Congestion notification can use the Link Layer Discovery Protocol (LLDP), defined in IEEE Std 802.1AB™, to advertise the CNPVs supported on a bridge's or end station's port, to determine whether the other bridges' or end stations' ports connected to the same LAN are configured for those same CNPVs, and to control the operation of the Congestion Notification Domain defenses (32.1.1).

Multiple instances of LLDP can be run on a single port. Each instance's LLDPDUs use a different destination_address, and therefore can reach different distances through a Virtual Bridged Network, depending on the types of the devices in that network and their connectivity. Since any bridge that is not congestion aware can spoil the ability of the Virtual Bridged Network to minimize frame loss, the selection of which LLDP instance is used for Automatic Congestion Notification Domain recognition defaults to the Nearest Bridge Address (01-80-C2-00-00-0E). This instance minimizes the reach of LLDP, and hence the risk that a non congestion aware bridge is present.

As described in 33.5, a single Congestion Notification TLV can be inserted into the LLDPDUs transmitted from any bridge or end station port that is configured to support a CNPV. The Congestion Notification TLV carries two fields that together provide for automatic control the of the Congestion Notification Domain defenses:

a) Eight Per-priority CNPV indicators (33.5.3), one per priority level, indicating whether each priority is or is not a CNPV on this port; and
b) Eight Per-priority Ready indicators (33.5.4), one per priority level, indicating whether the priority remap defenses are disabled on this port.

For every CNPV configured on a port, if LLDP finds exactly one remote neighbor, and if that neighbor's database in the port's LLDP table carries a Congestion Notification TLV (33.5) with that CNPV's bit set in the Per-priority CNPV indicators, then that CNPV is known to be configured on the remote system's port. This condition is summarized by the cnpdRcvdCnpv variable (32.4.11). Similarly, the cnpdRcvdReady variable (32.4.11) indicates whether or not the remote neighbor has or has not turned off its priority remapping defense. The read-only managed object cnpdAutoDefenseMode (32.4.3) indicates the results of the LLDP.

32.1.3 Variables controlling Congestion Notification Domain defense

In order to control CND defense, CN component managed objects (12.17.1), CN component priority managed objects (12.17.2), CN Port priority managed objects (12.17.3), and Congestion Point managed objects (12.17.4) override and are overridden by each other as follows:

a) Altering the CN component managed object (12.17.1) or CN component priority managed object (12.17.2) does not alter the values of any of the other managed objects (12.17.3, 12.17.4, 12.17.6).
b) If cngMasterEnable (32.2.1) is FALSE, all congestion notification activity is suppressed; the Priority Regeneration Table (6.9.3) operates normally, and CNMs, CN-TAGs, and LLDP Congestion Notification TLVs are never generated and are always ignored on receipt. If TRUE, the other managed objects control the operation of congestion notification.
c) The Port/priority and component objects override each other as shown in Table 32-1 and Table 32-2. In each case, if the Port/priority object contains the specified value, the component or component/ priority object controls the specified function.

Table 32-1—LLDP instance selection managed object overrides

(per-port per-priority) cnpdLldpInstanceChoice 32.4.4	(component per-priority) cncpLldpInstanceChoice 32.3.6	Congestion Notification TLV is sent/received	LLDP instance is selected by
cnlNone	any	No	none selected
cnlAdmin	any	Yes	(per-port per-priority) cnpdLldpInstanceSelector 32.4.5
cnlComponent	cnlNone	No	none selected
cnlComponent	cnlAdmin	Yes	(component per-priority) cncpLldpInstanceSelector 32.3.7

32.2 CN component variables

Every congestion aware end station or bridge component has a set of CN component variables to control the overall operation of congestion notification. All of these variables are included in the CN component managed object (12.17.1). Variables in this section marked, "CP only" are not required for an end station that does not support CPs. The CN component variables include:

Table 32-2—CND defense mode selection managed object overrides

(per-port per-priority) cnpdDefModeChoice 32.4.1	(component per-priority) cncpDefModeChoice 32.3.1	CND defense mode is selected by	Alternate priority is selected by
cpcAdmin	any	(per-port per-priority) cnpdAdminDefenseMode 32.4.2	(per-port per-priority) cnpdAlternatePriority 32.4.6
cpcAuto	any	(per-port per-priority) cnpdAutoDefenseMode 32.4.3	(component per-priority) cncpAutoAltPri 32.3.3
cpcComp	cpcAdmin	(component per-priority) cncpAdminDefenseMode 32.3.4	(component per-priority) cncpAlternatePriority 32.3.2
cpcComp	cpcAuto	(per-port per-priority) cnpdAutoDefenseMode 32.4.3	(component per-priority) cncpAutoAltPri 32.3.3

a) cngMasterEnable (32.2.1);
b) cngCnmTransmitPriority (32.2.2);
c) cngDiscardedFrames (32.2.3); and
d) cngErroredPortList (32.2.4).

32.2.1 cngMasterEnable

A Boolean value specifying whether congestion notification is enabled in this bridge component or end station.

32.2.2 cngCnmTransmitPriority

(CP only) The priority value to be used when transmitting CNMs from this bridge component or end station (default 6) (32.9.4:h).

NOTE—If cngCnmTransmitPriority is itself a CNPV, then it is possible that another CP could generate a CNM in response to a CNM sent by this Port. Although this is not desirable, the fact that CNMs are sent in response to only a sampling of frames means that the network will not fail due to an excessive number of CNMs.

32.2.3 cngDiscardedFrames

(CP only) The total number of frames discarded from full CP queues. This is the total of the values of all cpDiscardedFrames (32.8.12) variables for this end station or bridge component.

32.2.4 cngErroredPortList

(CP only) A list of Ports whose alternate priority values (cnpdAlternatePriority, 32.4.6) specify a CNPV, where 0 indicates that the bridge's alternate priority value (cncpAlternatePriority, 32.3.2) specifies a CNPV.

NOTE—The alternate priority (cncpAlternatePriority, 32.3.2, or cnpdAlternatePriority, 32.4.6) for a given CNPV cannot itself be a CNPV. But, because alternate priority values can be set using different managed objects for different CNPVs, invalid configurations can be specified. cngErroredPortList provides the network administrator with a list of Ports that could have invalid configurations.

32.3 Congestion notification per-CNPV variables

An instance of the Congestion notification per-CNPV variables exists for each CNPV in a congestion aware end station or bridge component to control Congestion notification for all Ports in the end station or bridge component. Variables in this section marked, "CP only" are not required for an end station that does not support CPs. The Congestion notification per-CNPV variables include:

a) cncpDefModeChoice (32.3.1);
b) cncpAlternatePriority (32.3.2);
c) cncpAutoAltPri (32.3.3);
d) cncpAdminDefenseMode (32.3.4);
e) cncpCreation (32.3.5);
f) cncpLldpInstanceChoice (32.3.6); and
g) cncpLldpInstanceSelector (32.3.7).

32.3.1 cncpDefModeChoice

An enumerated value specifying how the CND defense mode and alternate priority for all Ports for this CNPV are to be selected, unless overridden (32.1.3) by the variables in the CND defense per-Port per-CNPV variables (32.4), either:

1) **cpcAdmin:** The Port's default CND defense mode for this priority is controlled by cncpAdminDefenseMode (32.3.4) and the alternate priority by cncpAlternatePriority (32.3.2); or
2) **cpcAuto:** The Port's default CND defense mode for this priority is controlled by LLDP and the Congestion Notification TLV (cnpdAutoDefenseMode, 32.4.3) and the alternate priority by cncpAutoAltPri (32.3.3).

32.3.2 cncpAlternatePriority

(CP only) An alternate priority value to which this priority value is to be remapped when the Port's CND defense mode is cptEdge for this priority (default value 0). Can be overridden by cnpdAlternatePriority (32.1.3, 32.4.6).

32.3.3 cncpAutoAltPri

An integer indicating the next lower priority value than this CNPV that is not a CNPV in an end station or bridge component, or the next higher non-CNPV, if all lower values are CNPVs (32.1.1).

32.3.4 cncpAdminDefenseMode

An enumerated value controlling the CND defense mode (30.6) for all Ports for this CNPV in this bridge component or end station, either cptDisabled (1), cptInterior (2, the default value), cptInteriorReady (3), or cptEdge (4) (32.1.2). Can be overridden by cnpdAdminDefenseMode (32.1.3, 32.4.2).

32.3.5 cncpCreation

An enumerated value controlling the applicability of this CN component priority managed object to the Ports in that bridge component or end station, either:

1) **cncpAutoEnable:** The cnpdDefModeChoice variable (32.4.1) in any CN Port priority managed object created as a result of the creation of this CN component priority managed object or the creation of a Port is given the value cpcComp (3); or

2) **cncpAutoDisable:** The cnpdDefModeChoice variable (32.4.1) in any CN Port priority managed object created as a result of the creation of this CN component priority managed object or the creation of a Port is given the value cpcAdmin (1).

32.3.6 cncpLldpInstanceChoice

Determines the method by which the component LLDP instance selector, specifying on which LLDP instance the Congestion Notification TLV is to carry information for this priority, is determined (32.1.2), either.

1) **cnlNone:** if LLDP is not to carry Congestion Notification TLVs on any Port or priority; or
2) **cnlAdmin:** If cncpLldpInstanceSelector (32.3.7) governs LLDP instance selection for all Ports and priorities (this is the default value).

This choice can be overridden (32.1.3) on a per-Port per-priority basis by cnpdLldpInstanceChoice (32.4.4).

32.3.7 cncpLldpInstanceSelector

A reference to the LLDP destination address table entry whose LLDPDUs are to carry Congestion Notification TLVs for this bridge component or end station (32.1.2). The use of this variable is controlled by cncpLldpInstanceChoice (32.3.6).

32.4 CND defense per-Port per-CNPV variables

For each Port in a congestion aware end station or bridge component, and for each priority value for which a set of Congestion notification per-CNPV variables exists, there is an instance of the CND defense per-Port per-CNPV variables. These variables control congestion notification on the port for a CNPV, including Automatic Congestion Notification Domain recognition. Variables in this section marked, “CP only” are not required for an end station that does not support CPs. The CND defense per-Port per-CNPV variables include:

a) cnpdDefModeChoice (32.4.1);
b) cnpdAdminDefenseMode (32.4.2);
c) cnpdAdminDefenseMode (32.4.2);
d) cnpdAutoDefenseMode (32.4.3);
e) cnpdLldpInstanceChoice (32.4.4);
f) cnpdLldpInstanceSelector (32.4.5);
g) cnpdAlternatePriority (32.4.6);
h) cnpdXmitCnpvCapable (32.4.7);
i) cnpdXmitReady (32.4.8);
j) cncpDoesEdge (32.4.9);
k) cnpdAcceptsCnTag (32.4.10);
l) cnpdRcvdCnpv (32.4.11);
m) cnpdRcvdReady (32.4.12);
n) cnpdIsAdminDefMode (32.4.13); and
o) cnpdDefenseMode (32.4.14).

32.4.1 cnpdDefModeChoice

An enumerated value specifying how the CND defense mode and the alternate priority of the Port for this priority is to be determined, (30.6, 32.1.1, 32.1.2), either:

1) **cpcAdmin:** The Port's CND defense mode for this priority is controlled by cnpdAdminDefenseMode (32.4.2), and the alternate priority by cnpdAlternatePriority (32.4.6);
2) **cpcAuto:** The Port's CND defense mode for this priority is controlled by LLDP and the Congestion Notification TLV (cnpdAutoDefenseMode, 32.4.3) and the alternate priority by cncpAutoAltPri (32.3.3); or
3) **cpcComp:** The Port's CND defense mode for this priority is controlled by cncpDefModeChoice (32.3.1) and the alternate priority by cncpAutoAltPri (32.3.3) (this is the default value).

32.4.2 cnpdAdminDefenseMode

An enumerated value specifying the CND defense mode of the Port for this priority, if and only if cnpdDefModeChoice (32.4.1) has the value cpcAdmin (1) (30.6, 32.1.1, 32.1.2), either:

1) **cptDisabled:** The CP is disabled on this port for this CNPV. Priority regeneration on input is controlled by the priority regeneration table. This is the same behavior as when cngMasterEnable (32.2.1) has the value FALSE. (This is the default value.);
2) **cptInterior:** The priority parameter of frames input are not remapped to or from this priority, and CN-TAGs are not output;
3) **cptInteriorReady:** The priority parameter of frames input are not remapped to or from this priority, and CN-TAGs can be output;
4) **cptEdge:** The priority parameter of frames input at this priority are remapped to an alternate value, frames at other priorities are not remapped to this priority, and CN-TAGs are not output.

32.4.3 cnpdAutoDefenseMode

An enumerated value indicating the operating mode, either cptInterior (2), cptInteriorReady (3), cptEdge (4), in which the Port would operate for this CNPV if cnpdDefModeChoice (32.4.1) had the value cpcAuto (2).

NOTE—cncpAdminDefenseMode (32.3.4) is controlled by the network administrator. The object cnpdAutoDefenseMode (32.4.3) indicates what the CND defense mode would be, if it were controlled by the LLDP Congestion Notification TLV.

32.4.4 cnpdLldpInstanceChoice

Port and priority LLDP instance selector, specifying on which LLDP instance the Congestion Notification TLV is to carry information for this priority on this Port, either (32.1.2):

1) **cnlNone:** No LLDP Congestion Notification TLV is to carry Per-priority CNPV indicators or Per-priority Ready indicators on this Port for this priority;
2) **cnlAdmin:** cnpdLldpInstanceSelector (32.4.5) governs which LLDP instance is to carry Per-priority CNPV indicators and Per-priority Ready indicators for this priority in its Congestion Notification TLV on this Port; or
3) **cnlComponent:** cncpLldpInstanceSelector (32.3.7) governs LLDP instance selection for this Port and priority (this is the default value).

32.4.5 cnpdLldpInstanceSelector

A reference to the LLDP destination address table entry for an LLDP instance selector, specifying on which LLDP instance the Congestion Notification TLV is to carry information for this priority on this Port, if and only if cnpdLldpInstanceChoice (32.4.4) is set to cnlAdmin (2). Default value is 1. (32.1.2);

32.4.6 cnpdAlternatePriority

(CP only) An alternate priority value to which this priority value is to be remapped when the Port's CND defense mode is cptEdge for this priority, if and only if cnpdDefModeChoice (32.4.1) is set to cpcAdmin (1) (default value 0).

32.4.7 cnpdXmitCnpvCapable

Per-port per-priority Boolean variable indicating whether a given priority on this Port is (TRUE) or is not (FALSE) currently operating as a CNPV. cnpdXmitCnpvCapable is transmitted in the Per-priority CNPV indicators in the Congestion Notification TLV (33.5.3). The variable is TRUE if and only if all of the following conditions are met:

a) cngMasterEnable (32.2.1) is TRUE; and
b) The CND defense mode, as selected by cncpDefModeChoice, cncpAdminDefenseMode, cnpdDefModeChoice, and cnpdAdminDefenseMode, according to Table 32-2, is not cptDisabled.

32.4.8 cnpdXmitReady

Per-port per-priority Boolean variable indicating whether the priority remap defenses for this port and CNPV have been disabled. cnpdXmitReady is transmitted in the Per-priority Ready indicators in the Congestion Notification TLV (33.5.4). This variable is set and reset by the Congestion Notification Domain defense state machine (32.6).

32.4.9 cncpDoesEdge

Boolean variable indicating whether the cptEdge Congestion Notification Domain defense mode (32.1.1) is (TRUE) or is not (FALSE) implemented on this port for this priority. This variable is TRUE in a bridge. In an end station, it may be either TRUE or FALSE. This variable is not a managed object.

NOTE—This variable, when FALSE, causes the state machine in Figure 32-1 to pass through the CNDD_EDGE state directly to the CNDD_INTERIOR state. Since the state machine takes, in theory, no time to execute, it is impossible to tell from any measurement taken outside the system whether or not the actions specified in the CNDD_EDGE state are actually executed.

32.4.10 cnpdAcceptsCnTag

Boolean variable indicating whether the CN-TAG is (TRUE) or is not (FALSE) acceptable on data frames received on this port for this CNPV. This variable is TRUE in a bridge. In an end station, it may be either TRUE or FALSE. This variable is not a managed object.

NOTE 1—By preventing this Port and CNPV's cnpdXmitReady variable from being set TRUE, (see Figure 32-1) cnpdAcceptsCnTag prevents the adjacent device's Congestion Notification Domain defense state machine (32.6) from advancing from the CNDD_INTERIOR state to the CNDD_INTERIOR_READY state, and thus prevents it from transmitting a CN-TAG on this CNPV.

NOTE 2—This variable can be used by an end station that finds it convenient for the adjacent bridge to remove CN-TAGs from frames on a CNPV. However, since a CNM has a CN-TAG and can have a non-CNPV priority value, an end station can receive a CNM with a CN-TAG, even if this variable is FALSE.

32.4.11 cnpdRcvdCnpv

Per-port per-priority Boolean variable indicating the presence of an LLDP neighbor's Congestion Notification TLV (33.5). cnpdRcvdCnpv is TRUE only if all of the following are true:

a) An LLDP instance is selected by cncpLldpInstanceSelector (32.3.7) and cnpdLldpInstanceSelector (32.4.5);
b) The selected LLDP instance indicates that the Port has a single neighbor;
c) The selected LLDP instance's neighbor information includes a Congestion Notification TLV; and
d) That Congestion Notification TLV's Per-priority CNPV indicators field has a value of 1 in the position corresponding to this CNPV.

32.4.12 cnpdRcvdReady

Per-port per-priority Boolean variable indicating that the neighboring system has turned off its CND defenses. cnpdRcvdReady is TRUE only if all of the following are true:

a) cnpdRcvdCnpv is TRUE; and
b) The bit in the same Congestion Notification TLV's (33.5) Per-priority Ready indicators field (33.5.4) has a value of 1 in the position corresponding to this CNPV.

32.4.13 cnpdIsAdminDefMode

Boolean variable derived from the managed objects, indicating to the Congestion Notification Domain defense state machine (32.6) whether or not the CND defense mode is being forced to a particular state by the managed objects, as shown in Table 32-3.

Table 32-3—Determining cnpdIsAdminDefMode and cnpdDefenseMode

(per-port per-priority) cnpdDefModeChoice (32.4.1)	(component per-priority) cncpDefModeChoice (Clause 32)	cnpdIsAdminDefMode (32.4.13) is:	cnpdDefenseMode (32.4.14) is obtained from:
cpcAdmin	any	TRUE	(per-port per-priority) cnpdAdminDefenseMode 32.4.2
cpcAuto	any	FALSE	not defined
cpcComp	cpcAdmin	TRUE	(component per-priority) cncpAdminDefenseMode 32.3.4
cpcComp	cpcAuto	FALSE	not defined

NOTE—The purpose of this variable is to simplify the Boolean expressions in Figure 32-1.

32.4.14 cnpdDefenseMode

Enumerated value, indicating the management choice for the CND defense mode, if any, as shown in Table 32-3. The value of this variable is not defined if the managed variables indicate no choice for the CND defense mode.

NOTE—The purpose of this variable is to simplify the exit condition expressions in Figure 32-1.

32.5 Congestion Notification Domain defense procedures

There are three procedures associated with the Congestion Notification Domain defense state machine:

a) DisableCnpvRemapping() (32.5.1);
b) TurnOnCnDefenses() (32.5.2); and
c) TurnOffCnDefenses() (32.5.3).

32.5.1 DisableCnpvRemapping()

Removes any overrides by congestion notification of the Priority Regeneration Table (6.9.3).

32.5.2 TurnOnCnDefenses()

Overrides the Priority Regeneration Table (6.9.3) to use the alternate priority specified by cncpAlternatePriority (32.3.2), cncpAutoAltPri (32.3.3), and cnpdAlternatePriority (32.4.6) for frames with this CNPV, instead of the value in the Priority Regeneration Table.

32.5.3 TurnOffCnDefenses()

Overrides the Priority Regeneration Table (6.9.3) to not alter the priority of incoming frames with this CNPV, instead of using the value in the Priority Regeneration Table.

32.6 Congestion Notification Domain defense state machine

The Congestion Notification Domain defense state machine is illustrated in Figure 32-1. A congestion aware bridge component or end station shall implement one Congestion Notification Domain defense state machine per port per priority level.

NOTE—In Figure 32-1, global transitions are associated with each value of cnpdDefenseMode when the CND defense mode is chosen by the managed variables; the non-global transitions from state to state can take place only when the CND defense mode is chosen by LLDP.

32.7 Congestion notification protocol

Clause 32 is a detailed specification of the state machines required to implement the QCN algorithm introduced in 30.2. Table 32-4 shows the correlation between the description of QCN in 30.2, the state machine variables in this clause, and the CNM PDU fields in 33.4.

There are two stateful participants in the Congestion notification protocol:

a) The Congestion Point (31.1.1) shall behave in a manner that is indistinguishable, from an observer external to a system, from a strict implementation of the variables in 32.8 and the procedures in 32.9.
b) The Reaction Point (31.2.2.2) shall behave in a manner that is indistinguishable, from an observer external to a system, from a strict implementation of the timer in 32.12, variables in 32.13, the procedures in 32.14, and the state machine in 32.15.

32.8 Congestion Point variables

The following variables control the operation of a CP:

a) cpMacAddress (32.8.1);
b) cpId (32.8.2);
c) cpQSp (32.8.3);
d) cpQLen (32.8.4);

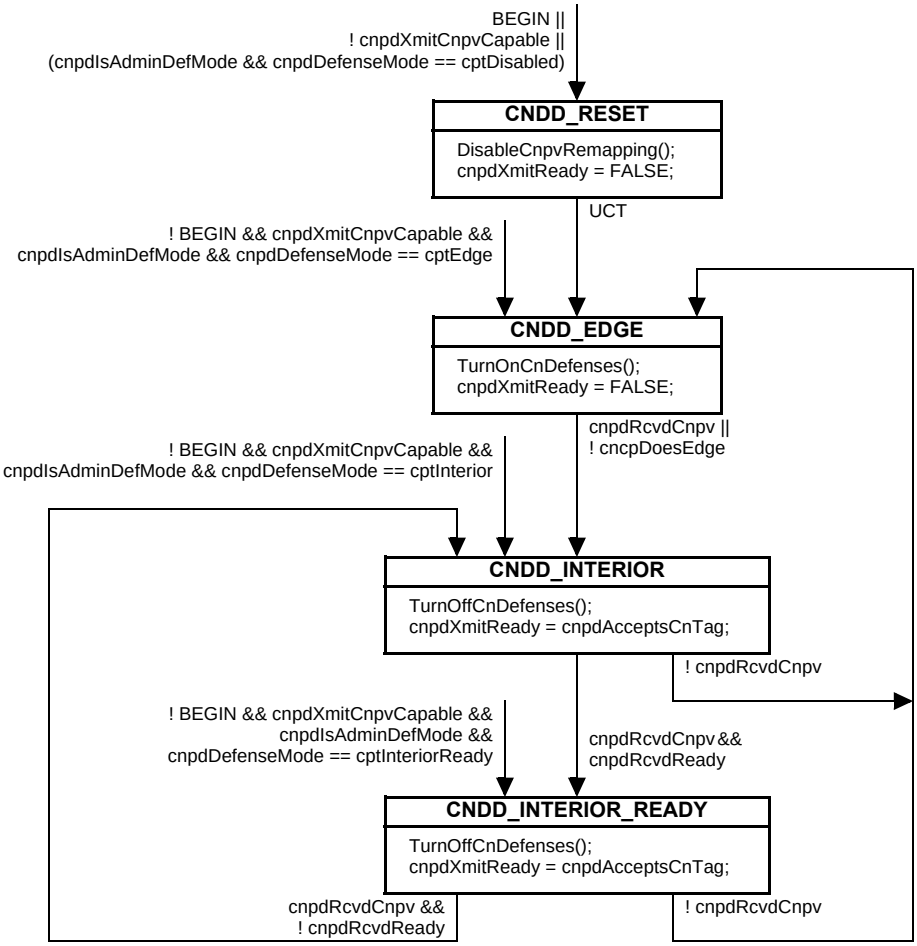

Figure 32-1—Congestion Notification Domain defense state machine

e) cpQLenOld (32.8.5);
f) cpW (32.8.6);
g) cpQOffset (32.8.7);
h) cpQDelta (32.8.8);
i) cpFb (32.8.9);
j) cpEnqued (32.8.10);
k) cpSampleBase (32.8.11);
l) cpDiscardedFrames (32.8.12);
m) cpTransmittedFrames (32.8.13);
n) cpTransmittedCnms (32.8.14); and
o) cpMinHeaderOctets (32.8.15).

32.8.1 cpMacAddress

The MAC address, belonging to the system transmitting the CNM PDU, used as the source_address of Congestion Notification Messages (CNMs) sent from this CP (32.9.4:b).

32.8.2 cpId

Unsigned integer. A number that, along with the source_address and vlan_identifier of a CNM PDU, uniquely identifies a CP in a Virtual Bridged Network (32.9.4:p).

Table 32-4—Correspondence of QCN protocol and CN message fields

Quantized Congestion Notification protocol, 30.2	Congestion notification protocol, 32.7	Congestion Notification Message, 33.3
Quantized Feedback calculation (32.8.9, 32.9.4:e)		
F_b (30.2.1)		cpFb (33.4.3)
Q_{off} (30.2.1)	− cpQOffset (32.8.7)	cnmQOffset (33.4.5)
Q (30.2.1)	cpQLen (32.8.4)	
Q_{eq} (30.2.1)	cpQSp (32.8.3)	
Q_{δ} (30.2.1)	cpQDelta (32.8.8)	cnmQDelta (33.4.6)
Q_{old} (30.2.1)	cpQLenOld (32.8.5)	
w (30.2.1)	cpW (32.8.6)	
RP rate calculations (32.14.5, 32.14.6, Figure 32-2)		
CR (30.2.2)	rpCurrentRate (32.13.6)	
TR (30.2.2)	rpTargetRate (32.13.5)	
Byte Counter (30.2.3)	rpByteCount (32.13.2)	
Timer (30.2.3)	RpWhile (32.12.1)	
G_d (30.2.2.1)	rpgGd (32.11.8)	
R_{AI} (30.2.2.1)	rpgAiRate (32.11.6)	
R_{HAI} (30.2.3)	rpgHaiRate (32.11.7)	
i (30.2.3)	min(rpByteStage, rpTimeStage) (RPR_HYPER_INCREASE)	

NOTE—This standard does not specify whether the CPID reported in a CNM by a CP that serves multiple CNPVs does or does not have the same value for its different CNPVs.

32.8.3 cpQSp

Unsigned integer. The set-point for the queue. This is the target number of octets in the CP's queue (32.9.4:e). Default value 26000.

32.8.4 cpQLen

Unsigned integer. The current number of octets in the CP's queue (32.9.4:c).

32.8.5 cpQLenOld

Unsigned integer. The previous value of cpQLen. cpQLenOld is updated from cpQLen each time GenerateCnmPdu() is called (32.9.4:c).

32.8.6 cpW

Real number. cpW is the weight to be given to the change in queue length in the calculation of cpFb (32.8.9). Default value 2. Although cpW is specified as a real number, it is constrained to be a power of 2, e.g. 1/2, 1, 2, 4, etc. In practice, therefore, it can be represented as a shift distance (plus an addition) in 32.9.4:e.

32.8.7 cpQOffset

The signed integer value of the transmitting CP's cpQSp (32.8.3) – cpQLen (32.8.4) used to calculate cpFb (32.8.9), and thus the Quantized Feedback field (33.4.3).

32.8.8 cpQDelta

The signed integer value of the transmitting CP's cpQLen (32.8.4) – cpQLenOld (32.8.5) used to calculate cpFb (32.8.9), and thus the Quantized Feedback field (33.4.3).

32.8.9 cpFb

Signed integer. Calculated just before the CP attempts to enqueue a frame:

cpFb = cpQOffset – cpW * cpQDelta, where:
cpQDelta = cpQLen – cpQLenOld and
cpQOffset = cpQSp – cpQLen

cpFb has two terms. The first term is the difference between the current and the desired queue lengths (cpQOffset, 32.8.7). The second is a weight factor cpW times the difference cpQDelta (32.8.8) between the current and the previous queue lengths. Thus, a multiple of the first derivative of the queue size is subtracted from the current non-optimality of the queue, so that if the queue length is moving toward the set point cpQSp, cpFb will be closer to 0 than if the queue length is moving away from cpQSp.

32.8.10 cpEnqued

Signed integer. The number of octets remaining to be enqueued by the CP before a CNM PDU is to be generated (32.9.3).

32.8.11 cpSampleBase

Unsigned integer. The minimum number of octets to enqueue in the CP's queue between CNM PDU transmissions (32.9.3). Default value 150,000.

32.8.12 cpDiscardedFrames

The number of frames offered to this CP that were discarded because of a full output queue (31.1.1).

32.8.13 cpTransmittedFrames

The number of data frames enqueued for transmission on this CP's output queue (31.1.1).

32.8.14 cpTransmittedCnms

The number of CNMs transmitted by this CP (32.9.4:s).

32.8.15 cpMinHeaderOctets

The minimum number of octets that the CP is to return in theEncapsulated MSDU field (33.4.10) of each CNM it generates (32.9.4:k). Default value 0.

32.9 Congestion Point procedures

The procedures used in a CP include:

a) Random() (32.9.1);
b) NewCpSampleBase() (32.9.2);
c) EM_UNITDATA.request (parameters) (32.9.3); and
d) GenerateCnmPdu() (32.9.4).

A CP needs no state machine; the procedure EM_UNITDATA.request (parameters), called each time a frame is presented for queuing, triggers all of the CP's functions.

32.9.1 Random

The Random function takes two parameter, min and max, and returns a pseudo-random number in the range min ≤ number < max. This function shall be initialized so that it generates a different value each time the system is reset.

32.9.2 NewCpSampleBase()

Called by EM_UNITDATA.request (parameters) (32.9.3) when generating a new value for cpSampleBase (32.8.11) from the CNM PDU Quantized Feedback field (33.4.3) last generated by GenerateCnmPdu(). NewCpSampleBase() returns a real number according to Table 32-5

Table 32-5—NewCpSampleBase() return value as a function of cpFb

Quantized Feedback / 8	value returned by NewCpSampleBase()
0	1
1	1/2
2	1/3
3	1/4
4	1/5
5	1/6
6	1/7
7	1/8

NOTE—The values in Table 32-5 are chosen so that the value returned by NewCpSampleBase() can be an integer to be used in a division, rather than a real number to be used in a multiplication.

32.9.3 EM_UNITDATA.request (parameters)

A CP offers an instance of the EISS (6.8) to the Queuing frames function (8.6.6). When called upon to enqueue a frame, the CP:

a) Determines the number of octets of buffer space required to enqueue the frame, and subtracts that number from cpEnqued (32.8.10);
b) If cpEnqued <= 0, then the CP:
    1) Calculates a new value for cpFb according to 32.8.9;
    2) Calls GenerateCnmPdu() to conditionally transmit a CNM PDU; and
    3) Replaces cpEnqued with cpSampleBase * NewCpSampleBase() * Random(0.85, 1.15), and if the result is less than zero, resets cpEnqued to 0.

The accuracy of the sample jitter introduced by the function Random(0.85, 1.15) is not critical to the success of congestion notification. The amount of jitter introduced by this function should be no more than $0.75 \leq$ jitter < 1.25 and no less than $0.875 \leq$ jitter < 1.125.

32.9.4 GenerateCnmPdu()

Called by the CP to conditionally generate a CN-TAGged CNM PDU for output. GenerateCnmPdu():

a) Uses the source_address of the frame triggering the CNM PDU as the destination_address of the generated CNM PDU;
b) Uses cpMacAddress (32.8.1) as the source_address of the CNM PDU;
c) Replaces the value of cpQLenOld (32.8.5) with cpQLen (32.8.4);
d) If cpFb is greater than or equal to 0, or if the destination_address parameter of the CNM PDU is a Group, and not an Individual, address, does nothing; no further processing takes place, and no CNM PDU is transmitted;
e) Fills the Quantized Feedback field (33.4.3) of the CNM PDU as follows:
    if (cpFb < – cpQSp * (2*cpW + 1))
        Quantized Feedback = 63
    else
        Quantized Feedback = – cpFb*63 / (cpQSp*(2*cpW+1))
f) If the Quantized Feedback field equal to 0 does nothing; no further processing takes place, and no CNM PDU is transmitted;
g) If the first octets of the mac_service_data_unit of the frame triggering the CNM PDU are a CN-TAG, copies the Flow Identifier field (33.2.1) from that CN-TAG to the CN-TAG of the CNM, else fills the Flow Identifier field of the CNM's CN-TAG with 0;
h) Sets the priority parameter of the CNM PDU from cngCnmTransmitPriority (32.2.2);

NOTE—The default value for the priority of a CNM PDU is 6. CNM PDUs are not expected to be sent in high volumes, but undelivered CNM PDUs reduce the ability of congestion notification to prevent lost data frames. Priority 7 is typically reserved for control traffic that, if not delivered, can result in the loss of connectivity in the bridged LAN.

i) Sets the vlan_identifier of the CNM PDU from the vlan_identifier of the frame triggering the CNM PDU, if the dot1agCfmVlanTable is not implemented, else the Primary VID of that vlan_identifier;
j) Fills the Encapsulated destination MAC address field (33.4.8) of the CNM PDU with the destination_address parameter of the frame triggering the CNM PDU;
k) Fills the Encapsulated MSDU field (33.4.10) of the CNM PDU with the first octets of the remainder of the mac_service_data_unit following the CN-TAG, if present, of the frame triggering the CNM PDU. The minimum number of octets is the value of cpMinHeaderOctets (32.8.15), or the whole of the mac_service_data_unit following the CN-TAG, if it is shorter, and the maximum is 64;
l) Fills the Encapsulated MSDU length field (33.4.9) of the CNM PDU with the number of octets inserted into the Encapsulated MSDU field (33.4.10);

m) Inserts the encapsulation appropriate to the CP's port, as specified in 33.3, at the beginning of the mac_service_data_unit;

NOTE 3—This encapsulation is changed by the bridge, if required, before the CNM PDU is output on a port that uses a different encapsulation from the CP's port, as described in Clause 5 of IEEE Std 802.1H.

n) Fills the Version field (33.4.1) of the CNM PDU with 0;
o) Fills the ReservedV field (33.4.2) of the CNM PDU with 0;
p) Fills the Congestion Point Identifier field (33.4.4) of the CNM PDU from the variable cpId (32.8.2);
q) Computes and fills the cnmQOffset and cnmQDelta fields (33.4.5, 33.4.6) from the variables cpQOffset and cpQDelta;
r) Fills the Encapsulated priority field (33.4.7) from the priority parameter of the frame triggering the CNM PDU;
s) Increments cpTransmittedCnms (32.8.14); and
t) Passes the CN-TAG and CNM PDU as an EM_UNITDATA.indication to the higher layers (Figure 31-1), if the CP is in a bridge, or to the Flow multiplexer (31.2.4), if the CP is in an end station (Figure 31-2).

32.10 Reaction Point per-Port per-CNPV variables

There is one set of these variables per Port per CNPV in a congestion aware end station. A set of Reaction Point per-Port per-CNPV variables is created or deleted whenever a CN Port priority managed object is created or deleted. All of the RPs (if more than one) associated with a given Port and CNPV share a single set of the Reaction Point per-Port per-CNPV variables.

32.10.1 rpppMaxRps

An unsigned integer controlling the maximum number of RPs allowed for this CNPV on this Port (default 1). An end station shall not create more than rpppMaxRps on a given Port, but it can create fewer.

32.10.2 rpppCreatedRps

The number of times an RP's rpEnabled variable has been set TRUE at this priority level on this Port (32.14.4:e).

32.10.3 rpppRpCentiseconds

An integer containing the number of RP-centiseconds accumulated by RPs at this priority level on this Port. This variable is incremented once per centisecond (10 ms) for each RP on this Port whose rpEnabled variable (32.13.1) is TRUE.

NOTE—Reading rpppRpCentiseconds and SysUpTime[1] at two different times allows a management entity to compute the average number of RPs present for a priority and Port by dividing the change in rpppRpCentiseconds by the change in SysUpTime (which is measured in centiseconds).

32.11 Reaction Point group variables

There is one set of Reaction Point group variables for each group of RPs belonging to a particular CNPV on a Port in a congestion aware end station. It is an implementation choice whether each RP rate control state machine (32.15) has its own set of Reaction Point group variables or shares them with another RP rate

1. From IETF RFC 3418 (STD 62) "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)".

control state machine on the same Port and the same CNPV. This standard does not specify how RPs are grouped together for management purposes, except that all of the RPs controlled by a single set of Reaction Point group variables shall serve a single CNPV. At the extremes, a single set of Reaction Point group variables could control all of the RPs serving the same CNPV, or each RP could be controlled by its own set of Reaction Point group variables. The Reaction Point group variables include:

a) rpgEnable (32.11.1);
b) rpgTimeReset (32.11.2);
c) rpgByteReset (32.11.3);
d) rpgThreshold (32.11.4);
e) rpgMaxRate (32.11.5);
f) rpgAiRate (32.11.6);
g) rpgHaiRate (32.11.7);
h) rpgGd (32.11.8);
i) rpgMinDecFac (32.11.9); and
j) rpgMinRate (32.11.10).

32.11.1 rpgEnable

A Boolean value controlling the rpEnabled variables for all of the RP rate control state machines controlled by this Reaction Point group managed object. If rpgEnable is TRUE, the controlled rpEnabled variables are not held in the FALSE state, thus enabling the RPs to pay attention to received CNMs. If rpgEnable is FALSE, the controlled rpEnabled variables are held in the FALSE state, thus disabling the RPs.

32.11.2 rpgTimeReset

Unsigned integer. RpWhile (32.12.1) is reset to either rpgTimeReset or rpgTimeReset / 2 (with random jitter—see Figure 32-2, RPR_ACTIVE_TIME) each time it reaches 0, according to the state of the RP rate control state machine. Default value 15 ms.

32.11.3 rpgByteReset

Unsigned integer. rpByteCount (32.13.2) is reset to either rpgByteReset or rpgByteReset / 2 (with random jitter—see Figure 32-2, RPR_ACTIVE_BYTE) each time it reaches 0, according to the state of the RP rate control state machine. Default value 150 Koctets.

32.11.4 rpgThreshold

Unsigned integer. Specifies the number of times rpByteStage or rpTimeStage can count before the RP rate control state machine advances states. Default value is 5.

32.11.5 rpgMaxRate

Unsigned integer. The maximum rate, in bits per second, at which an RP can transmit. Default value is the speed of the Port. A system shall support a non-0 minimum value for rpgMaxRate no larger than 5 Mbit/s. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc. rpgMaxRate is configurable in multiples of 1 Mbit/s.

32.11.6 rpgAiRate

Unsigned integer. The rate, in bits per second, used to increase rpTargetRate in the RPR_ACTIVE_INCREASE state in Figure 32-2. Default value 5 Mbit/s. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc. rpgAiRate is configurable in multiples of 1 Mbit/s.

32.11.7 rpgHaiRate

Unsigned integer. The rate, in bits per second, used to increase rpTargetRate in the RPR_HYPER_INCREASE state in Figure 32-2. Default value 50 Mbit/s. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc. rpgHaiRate is configurable in multiples of 1 Mbit/s.

32.11.8 rpgGd

Real number. Multiplied times the Quantized Feedback field (33.4.3) received in a CNM PDU to decrease rpTargetRate. Default value 1/128. rpgGd is configurable as a power of 2, e.g. 1/256, 1/128, 1/64, etc.

32.11.9 rpgMinDecFac

Real number. The minimum factor by which the current RP transmit rate rpCurrentRate can be changed by reception of a CNM. Default value is 0.5.

32.11.10 rpgMinRate

Unsigned integer. The minimum value, in bits per second, for rpCurrentRate (32.13.6) Default value is 10 Mbit/s. A system shall support a minimum value for rpgMinRate no larger than 10 Mbit/s. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc. rpgMinDecFac is configurable in multiples of 1 Mbit/s.

32.12 Reaction Point timer

An RP implements one timer:

a) RpWhile (32.12.1).

32.12.1 RpWhile

Timer counter for controlling the RP rate control state machine (32.15). RpWhile operates normally when rpFreeze (32.13.7) is FALSE, and shall not be decremented when rpFreeze is TRUE.

32.13 Reaction Point variables

Each RP rate control state machine (32.15) has its own set of Reaction Point variables. The Reaction Point variables include:

a) rpEnabled (32.13.1);
b) rpByteCount (32.13.2);
c) rpByteStage (32.13.3);
d) rpTimeStage (32.13.4);
e) rpTargetRate (32.13.5);
f) rpCurrentRate (32.13.6);
g) rpFreeze (32.13.7);
h) rpLimiterRate (32.13.8); and
i) rpFb (32.13.9).

32.13.1 rpEnabled

Boolean. Controls RP rate control state machine. If TRUE, state machine is running. If FALSE, state machine is held in the RPR_RESET state. Initialized to FALSE when an RP rate control state machine is created. Held in the FALSE state if rpgEnable (32.11.1) has the value FALSE.

32.13.2 rpByteCount

Unsigned integer. Counts octets passed from the Rate Limiter (31.2.2.4) to the Flow Selection function (31.2.2.5). The counting down of rpByteCount to or past 0 triggers a reevaluation of rpCurrentRate (32.13.6).

32.13.3 rpByteStage

Unsigned integer. Counts the number of times rpByteCount has counted down to or past 0 since the last CNM was received.

32.13.4 rpTimeStage

Unsigned integer. Counts the number of times RpWhile has counted down to 0 since the last CNM was received.

32.13.5 rpTargetRate

Unsigned integer. The target rate, in bits per second, toward which rpCurrentRate is heading. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc.

32.13.6 rpCurrentRate

Unsigned integer. The rate, in bits per second, at which the Rate Limiter (31.2.2.4) transmits data from its Flow queue when rpFreeze (32.13.7) is FALSE. (The actual output of the Rate Limiter is controlled by rpLimiterRate, 32.13.8.) This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc.

32.13.7 rpFreeze

Boolean. When FALSE, the RP and Rate Limiter operate normally. When TRUE, rpLimiterRate is held to the value 0, to suppress the transmission of frames. rpFreeze is FALSE when there is room for frames from this RP in the output queue, and TRUE when not.

32.13.8 rpLimiterRate

Unsigned integer. The rate, in bits per second, at which the Rate Limiter (31.2.2.4) transmits data from its Flow queue. rpLimiterRate is equal to rpCurrentRate (32.13.6) as long as rpFreeze (32.13.7) is FALSE, and 0, when rpFreeze is TRUE. This rate includes all bits consequent to transmitting the frame on the LAN, including preamble, inter-frame gap, etc.

32.13.9 rpFb

Unsigned integer. The value last received in the Quantized Feedback field (33.4.3) of a CNM PDU or 0.

32.14 Reaction Point procedures

The following procedures are used by the RP rate control state machine and the Output flow segregation function (31.2.1):

a) ResetCnm (32.14.1);
b) TestRpTerminate (32.14.2);
c) TransmitDataFrame (32.14.3);
d) ReceiveCnm (32.14.4);
e) ProcessCnm (32.14.5); and
f) AdjustRates (32.14.6).

32.14.1 ResetCnm

Called whenever the RP rate control state machine enters the RPR_RESET state (Figure 32-2) to set the following variables:

```
rpCurrentRate = rpgMaxRate;
rpTargetRate  = rpgMaxRate;
rpByteCount   = rpgByteReset;
rpFb          = 0;
```

32.14.2 TestRpTerminate

Called by the Flow Selection function (31.2.2.5) each time a frame is passed from the Per-CNPV station function's Rate Limiter (31.2.2.4) to the Flow multiplexer (31.2.4). If:

a) rpCurrentRate equals rpgMaxRate;
b) The Flow queue is empty; and
c) rpEnabled is TRUE;

Then TestRpTerminate sets rpEnabled (32.13.1) to FALSE, which disables the RP until another CNM is received.

NOTE—Condition (b) can be met when all frames limited by a given Rate Limiter have been processed. See 31.2.2.1.

32.14.3 TransmitDataFrame

TransmitDataFrame() is called each time a frame is passed from a Rate Limiter (31.2.2.4) to the Port's Flow multiplexer (31.2.4). TransmitDataFrame() subtracts the length of the transmitted frame from rpByteCount (32.13.2) and may insert a Congestion Notification Tag (CN-TAG) containing the Flow ID of the Flow queue at the beginning of the frame's mac_service_data_unit.

If an end station is VLAN-aware or priority-aware, and if a CN-TAG is added to a frame, the CN-TAG shall follow the VLAN or priority tag in the frame.

32.14.4 ReceiveCnm

Called whenever a CNM is received. ReceiveCnm() performs the following actions:

a) The CNM is validated according to 33.4.11 and is discarded if invalid;
b) If the CNM frame carries a CN-TAG (33.2), the Flow Identifier from that CN-TAG (33.2.1) is used to identify the particular RP rate control state machine to which this CNM applies;

c) If the receiving end station is unable to identify the particular RP rate control state machine to which this CNM applies, it discards the CNM, and no further processing takes place;
d) Sets the rpFb variable from the CNM's Quantized Feedback field (33.4.3); and
e) If the selected RP rate control state machine's rpEnabled variable is FALSE and is not held FALSE because rpgEnable (32.11.1) has the value FALSE, and the CNM's cnmQOffset field (33.4.5) is negative, then rpEnabled is reset to TRUE, the variable rpppCreatedRps (32.10.2) is incremented.

These actions activate the selected RP rate control state machine, if it was not active (rpEnabled == FALSE), and also cause the RP rate control state machine to process the received CNM.

32.14.5 ProcessCnm

Called whenever the RP rate control state machine enters the RPR_CNM_RECEIVED state (Figure 32-2) to perform the following actions:

```
if (rpByteStage ! = 0) {
        rpTargetRate = rpCurrentRate;
        rpByteCount = rpgByteReset;
}
rpByteStage = 0;
rpTimeStage = 0;
dec_factor = (1 - (rpgGd * rpFb));
if (dec_factor < rpgMinDecFac)
        dec_factor = rpgMinDecFac;
rpCurrentRate = rpCurrentRate * dec_factor;
if (rpCurrentRate < rpgMinRate)
        rpCurrentRate = rpgMinRate;
RpWhile = rpgTimeReset;
```

32.14.6 AdjustRates

AdjustRates is called whenever the RPR_ADJUST_RATES, RPR_ACTIVE_INCREASE, or RPR_HYPER_INCREASE states are entered. It takes one parameter, *increase*, that specifies how much to increase rpTargetRate, and performs the following actions:

```
if ((rpByteStage == 1) || (rpTimeStage == 1))   &&    (rpTargetRate > 10 * rpCurrentRate))
        rpTargetRate = rpTargetRate / 8;
else
        rpTargetRate = rpTargetRate + increase;
rpCurrentRate = (rpTargetRate + rpCurrentRate)/2;
if (rpCurrentRate > rpgMaxRate)
        rpCurrentRate = rpgMaxRate;
```

32.15 RP rate control state machine

The RP rate control state machine is illustrated in Figure 32-2. In that figure, the RPR_RESET state has no exit. The state machine is held in the RPR_RESET state as long as either the general system initialization variable "BEGIN" is TRUE or the rpEnabled variable, local to the particular instance of the RP rate control state machine, is FALSE. When "BEGIN" becomes FALSE and rpEnabled TRUE, the state machine remains in the RPR_RESET state until a CNM is received by ReceiveCnm (32.14.4) with a Quantized Feedback field (33.4.3) that is less than 0. ReceiveCnm places that field into rpFb (32.13.9), which condition forces the state machine into the RPR_CNM_RECEIVED state. When that state resets rpFb to 0, progress through the state machine continues until either another CNM is received and rpFb is set to a non-zero value,

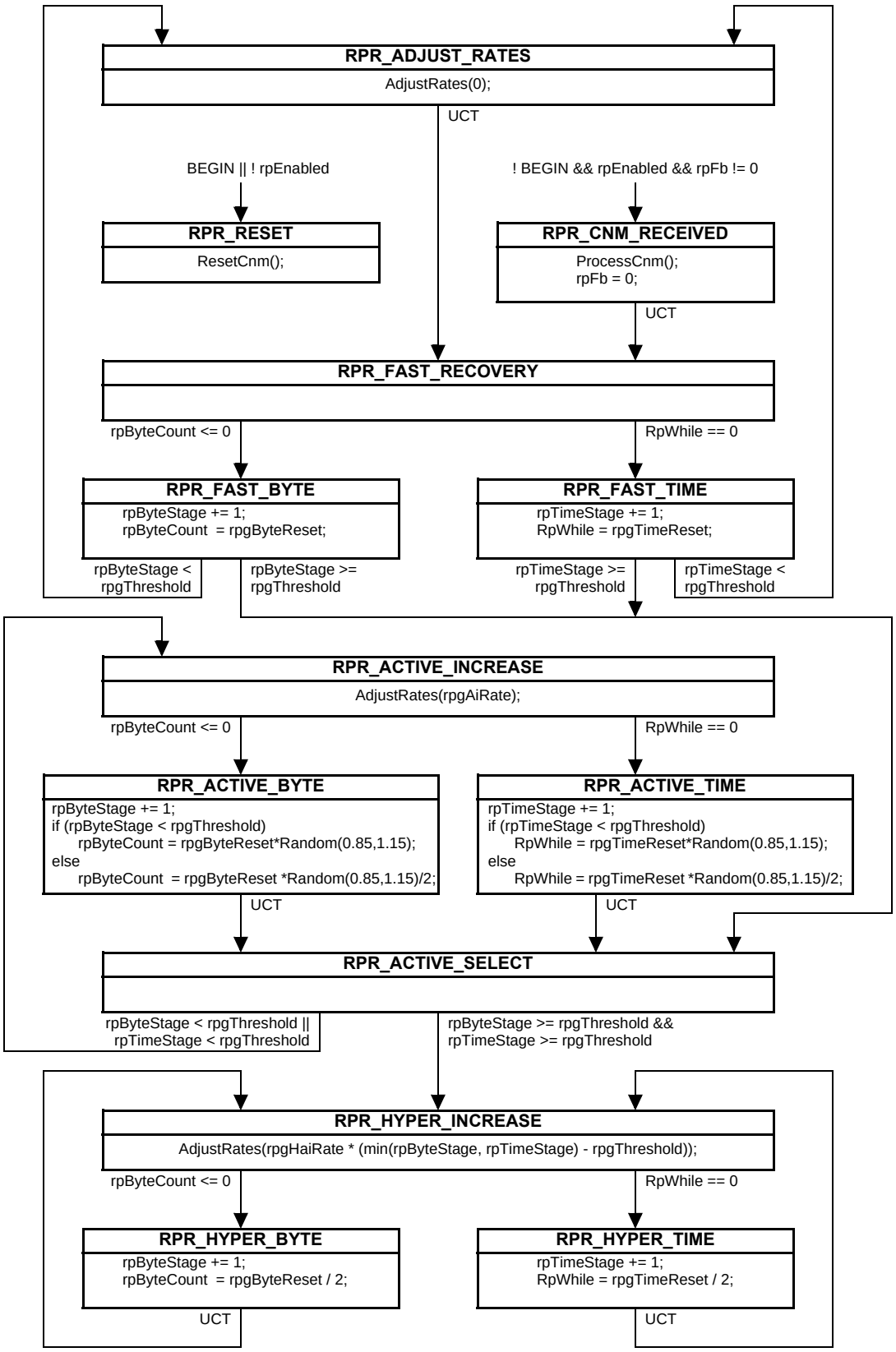

Figure 32-2—RP rate control state machine

or until the state machine is deactivated by TestRpTerminate (32.14.2) setting rpEnabled to FALSE.

32.16 Congestion notification and encapsulation interworking function

As mentioned in 30.8, a hierarchical Bridged Network, e.g., a Provider Backbone Bridged Network, can require a Congestion notification and encapsulation interworking function for a CP in the core of the network to generate a CNM that can reach the RP, which is presumably outside the core. This general case is illustrated in Figure 32-3. In the network illustrated, encapsulation and decapsulation functions peer with each other to encapsulate the data frames inside some kind of wrapper, e.g., an S-TAG and/or an I-TAG.

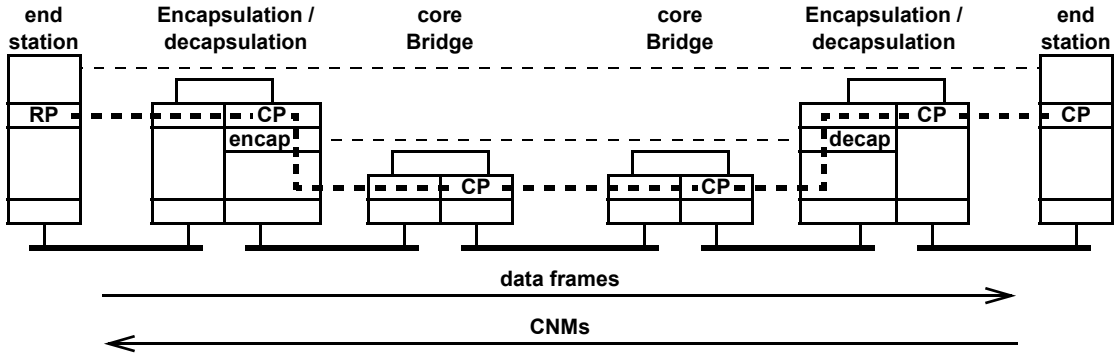

Figure 32-3—CP–RP peering in any hierarchical Bridged Network

Not all encapsulation schemes are equivalent. It is useful to divide hierarchical Bridged Networks into two categories for the purpose of defining Congestion notification and encapsulation interworking functions, based on the mac_service_data_unit of a data frame carrying (optionally) a CN-TAG as seen by a CP residing in a core Bridge, with one of these cases further subdivided:

a) In a network where the original end station's mac_service_data_unit, including the optional CN-TAG, comprises the first octets of the mac_service_data_unit at the core Bridge's CP, there is no need for a Congestion notification and encapsulation interworking function. For example, a Provider Bridged network in which the C-TAGs are not carried across the core needs no interworking function.

b) In a network where an encapsulation comprises the first octets of the mac_service_data_unit at the core Bridge's CP, a Congestion notification and encapsulation interworking function is required. It is useful to make a further distinction, in this case:

   1) The MAC addresses of Ports in the Bridges where the encapsulation and decapsulation functions are performed are the destination and source addresses of frames crossing the core Bridges. For example, in a Provider Backbone Bridged Network, the I-TAG, and perhaps an S-TAG and/or a C-TAG as well, can be present as the first octets of the mac_service_data_unit, and the source_address and destination_address parameters visible to the core Bridge's CP are the addresses of PIPs in Backbone Edge Bridges.

   2) In other networks, although an encapsulation is in the first octets of the mac_service_data_unit at the core Bridge's CP, the source and destination MAC addresses are not altered by the encapsulation and decapsulation operations. For example, in a Provider Bridged network in which both a C-TAG and an S-TAG are carried across the core, the C-TAG occupies the first octets of the mac_service_data_unit as seen by a core Bridge's CP.

In either type of network covered by case b, the core Bridge's CP is configured by the network administrator to return, in the Encapsulated MSDU field of the CNM PDU (33.4.10), the leading octets of the mac_service_data_unit of the data frame triggering transmission of the CNM. Since the CN-TAG of the data frame, if present, is buried behind encapsulations unknown to the CP, when the CN-TAG and CNM are built

according to 32.9.4, the Flow Identifier returned in the CN-TAG of the CNM is filled with 0. The network administrator configures the CP to return at least enough octets of the data frame's mac_service_data_unit to include the CN-TAG of the original data frame transmitted by the RP.

The difference between case 1 and case 2, preceding, lies in the means used by the Port containing the Congestion notification and encapsulation interworking function to recognize the CNM in order to translate it. In case 1, the CNM is addressed to that Port, and has a CN-TAG and the CNM encapsulation, instead of a data frame tag and encapsulation. In case 2, the only reliable indicator of a CNM that needs to be transformed, as opposed for example, to a CNM returned by the destination end station's CP, can be the presence of an encapsulation known to the interworking function in the Encapsulated MSDU field (33.4.10) of the CNM PDU.

When the CNM generated by the core Bridge's CP reaches a Congestion notification and encapsulation interworking function, that interworking function uses the information in the CNM PDU, including the encapsulated data frame information, to construct a CN-TAG and CNM PDU to send to the RP. The steps taken are:

c) In case 1, where the CNM is returned to the MAC address of the interworking function's Port, the destination_address parameter is obtained from the appropriate source address found in the Encapsulated MSDU field, and the source_address parameter is a MAC address belonging to the Bridge containing the interworking function, and valid in the context of the newly generated CNM, for example, a management port address;
d) In case 2, where the CNM is recognized by the presence of the encapsulation in the Encapsulated MSDU field, the source_address, and destination_address parameters are unchanged;
e) The vlan_identifier is obtained from the Encapsulated MSDU field;
f) The drop_eligible parameter is False;
g) The priority parameter used is the value configured for a CP on a Port in the Bridge in which the interworking function resides, and through which the CNM passes;
h) The Encapsulated priority field (33.4.7) is obtained from the Encapsulated MSDU field, or if not present there, the Encapsulated priority is left unchanged;

NOTE—In spite of the provisions for CND defense in this clause, it is possible for a network administrator to configure a network so that the priority of a CN data frame is not preserved. For example, in a Provider Bridged Network, two CNPVs could be mapped to the same S-TAG priority, and the C-TAG not configured to be carried across the network. In that case, an end station with an RP on each of those CNPVs, and that does not distinguish between those RPs using CN-TAGs, would be unable to assign returned CNMs to the right RP. At least some of its CCFs would not be constrained, and uncontrolled congestion would likely result. It is therefore recommended that the CN-TAG be included across a Provider Bridged Network, or that each CNPV be mapped to a separate priority in the backbone.

i) If a CN-TAG is found in the Encapsulated MSDU, its Flow Identifier is copied to the CN-TAG of the newly generated CNM, else 0 is used for that CN-TAG's Flow Identifier; and
j) The encapsulation in the Encapsulated MSDU field of the CNM PDU, up to and including the CN-TAG (which is not necessarily present), is elided, the remainder (if any) of the Encapsulated MSDU field moved up to the first octets of the field, and the Encapsulated MSDU length field reduced accordingly.

Insert a new Clause 33 as follows.

33. Encoding of congestion notification Protocol Data Units

This clause specifies the method of encoding congestion notification PDUs. The specifications include:

a) The general structure and encoding of congestion notification PDUs (33.1);
b) The format used for the Congestion Notification Tag (CN-TAG, 33.2);
c) The encapsulation used for a Congestion Notification Message (CNM, 33.3);
d) The format used for the CNM (33.4); and
e) The format of the IEEE Std. 802.1AB-2004 Type Length Value (TLV) used to signal that a VLAN-aware Bridge or end station is congestion aware, and the state of its Congestion Notification Domain defense mode (33.5).

NOTE—Clause 30 introduces the principles of congestion notification operation and the network architectural concepts that support it. Clause 31 breaks down the congestion notification entities into their components. Clause 32 specifies the protocols operated by these components.

33.1 Structure, representation, and encoding

All congestion notification PDUs contain an integral number of octets.

The octets in a congestion notification PDU are numbered starting from 1 and increasing in the order they are put into the MAC Service Data Unit (MSDU) that accompanies a request to or indication from the instance of the MAC Internal Sublayer Service (ISS or EISS) used by a congestion notification entity.

The bits in an octet are numbered from 1 to 8 in order of increasing bit significance, where 1 is the least significant bit in the octet.

Where octets and bits within a congestion notification PDU are represented using a diagram, octets shown higher on the page than subsequent octets and octets shown to the left of subsequent octets at the same height on the page are lower numbered; bits shown to the left of other bits within the same octet are higher numbered.

Where two or more consecutive octets are represented as hexadecimal values, lower numbered octet(s) are shown to the left and each octet following the first is preceded by a hyphen, e.g., 01-80-C2-00-00-00.

When consecutive octets are used to encode a binary number, the lower octet number has the more significant value. When consecutive bits within an octet are used to encode a binary number, the higher bit number has the most significant value. When bits within consecutive octets are used to encode a binary number, the lower octet number composes the more significant bits of the number. A flag is encoded as a single bit, and is set (TRUE) if the bit takes the value 1, and clear (FALSE) otherwise. The remaining bits within the octet can be used to encode other protocol fields.

33.2 Congestion Notification Tag format

The means for identifying Congestion Notification Tag PDUs depend on the medium. For media using a Type/Length field, e.g., IEEE 802.3 media, the identification consists of two octets containing the Type value shown (in hexadecimal notation) in Table 33-3. Media requiring an LLC encapsulation (e.g., IEEE 802.11) use the SNAP encoding shown (in hexadecimal notation) in Table 33-4.

Table 33-1—Congestion Notification Tag Encapsulation: Type/Length Media

	Octet	Length
Tag type (To Be Determined)	1	2
Flow Identifier	3	2

Table 33-2—Congestion Notification Tag Encapsulation: LLC Media

	Octet	Length
0xAAAA03	1	3
0x000000	4	3
Tag type (To Be Determined)	7	2
Flow Identifier	9	2

In a VLAN-aware Bridge, the shim that supports the Enhanced Internal Sublayer Service (6.1, 6.8, also see Figure 22-4) is below the Queuing shim (8.6.5, 8.6.6, 8.6.7, 8.6.8, 31.1.1). The CN-TAG is examined by the CP, which is part of the Queuing shim. Therefore, the CN-TAG will be further from the source_address and destination_address fields, and closer to the end of the frame, than a VLAN tag added by 6.8, if any. In order for a bridge to be able to recognize the CN-TAG in a data frame, an end station shall insert the CN-TAG and VLAN tag in the relative order required by a bridge: addresses, VLAN tag, CN-TAG, data[1].

33.2.1 Flow Identifier

The meaning and encoding of the Flow Identifier field are not specified by this standard. A CP shall not interpret the value in a Flow Identifier field. The Flow Identifier returned in the CN-TAG of a CNM is used by an end station's CNM demultiplexer (31.2.5) to identify the RP, the Flow queue, or a group of one or more individual flows, from which was transmitted the data frame that triggered the CNM.

33.3 Congestion Notification Message

The means for identifying Congestion Notification Message PDUs depend on the medium. For media using a Type/Length field, e.g., IEEE 802.3 media, the identification consists of two octets containing the Type value shown (in hexadecimal notation) in Table 33-3. Media requiring an LLC encapsulation (e.g., IEEE 802.11) use the SNAP encoding shown (in hexadecimal notation) in Table 33-4.

Table 33-3—Congestion Notification Message Encapsulation: Type/Length Media

	Octet	Length
PDU type (0x22E7)	1	2
CNM PDU	3	16

1. There are other tags, e.g., the IEEE 802.1AE MAC security tag, not included in this abbreviated list.

Table 33-4—Congestion Notification Message Encapsulation: LLC Media

	Octet	Length
0xAAAA03	1	3
0x000000	4	3
PDU type (0x22E7)	7	2
CNM PDU	9	16

33.4 Congestion Notification Message PDU format

The format of a Congestion Notification Message PDU is illustrated in Table 33-5. If the frame that triggered the transmission of the CNM carried a CN-TAG, then so does the frame carrying the CNM PDU.

Table 33-5—Congestion Notification Message PDU

	Octet	Length
Version	1	4 bits
ReservedV	1, 2	6 bits
Quantized Feedback	2	6 bits
Congestion Point Identifier (CPID)	3	8
cnmQOffset	11	2
cnmQDelta	13	2
Encapsulated priority	15	2
Encapsulated destination MAC address	17	6
Encapsulated MSDU length	23	2
Encapsulated MSDU	25	0—64

33.4.1 Version

This field, 4 bits in length, shall be transmitted with the value 0, and shall be ignored on receipt. The Version field occupies the most significant bits of the first octet of the CNM PDU.

33.4.2 ReservedV

This field, 6 bits in length, shall be transmitted with the value 0, and shall be ignored on receipt. The ReservedV field occupies the least-significant 4 bits of the first octet, and the most significant 2 bits of the second octet, of the CNM PDU.

33.4.3 Quantized Feedback

This field, 6 bits in length, contains the Quantized Feedback value calculated by the Congestion Point's EM_UNITDATA.request (parameters) procedure (32.9.3). The Quantized Feedback field occupies the least significant bits of the first two octets of the CNM PDU.

33.4.4 Congestion Point Identifier

This field, 8 octets in length, uniquely identifies the CP that triggered the transmission of this Congestion Notification Message PDU format within the Congestion Notification Domain. Because the transmitting system can format the CPID in any manner, a receiving RP shall not assign any meaning to subfields within a CPID.

NOTE—Making the CPID 8 octets long removes the connection between CP identity and the source MAC address of the CNM PDU. One way to achieve the requirements for uniqueness of the CPID would be to use a Port's MAC address as the high-order 6 octets of the CPID, and use the low-order two octets of the CPID to hold the priority of the CP's queue and/or an indication of the physical port.

33.4.5 cnmQOffset

The 2's-complement signed integer value of the transmitting CP's cpQOffset (32.8.7) in units of 64 octets. This standard does not specify whether cpQOffset is rounded or truncated to obtain cnmQOffset. If the value of cpQOffset is less than –32768, a value of –32768 is used in cpQOffset, and if greater than 32767, 32767 is used.

33.4.6 cnmQDelta

The 2's-complement signed integer value of the transmitting CP's cpQDelta (32.8.8) in units of 64 octets. This standard does not specify whether cpQDelta is rounded or truncated to obtain cnmQDelta. If the value of cpQDelta is less than –32768, a value of –32768 is used in cnmQDelta, and if greater than 32767, 32767 is used.

NOTE—Although cnmQDelta is not used by the RP, it is included in the CNM PDU to aid the network administrator when adjusting configuration parameters.

33.4.7 Encapsulated priority

This field, 2 octets in length, contains the priority parameter of the frame that triggered the transmission of this Congestion Notification Message in the most significant 3 bits of the field. The remaining bits of the field are transmitted as 0, and ignored on receipt.

33.4.8 Encapsulated destination MAC address

This field, 6 octets in length, contains the destination_mac_address parameter of the frame that triggered the transmission of this Congestion Notification Message.

33.4.9 Encapsulated MSDU length

This field, 2 octets in length, contains the number of octets returned in the Encapsulated MSDU field (33.4.10).

33.4.10 Encapsulated MSDU

This field, a maximum of 64 octets in length, contains the initial octets of the mac_service_data_unit of the frame that triggered the transmission of this Congestion Notification Message.

33.4.11 CNM Validation

A CNM PDU received by a CNM demultiplexer (31.2.5) shall be considered invalid and be discarded if the following condition is TRUE:

a) There are fewer than 24 octets in the mac_service_data_unit.

A CNM PDU received by a CNM demultiplexer (31.2.5) may be considered invalid and be discarded if the following condition is TRUE:

b) The CNM PDU has no CN-TAG.

NOTE 1—The variable cnpdAcceptsCnTag (32.4.10), when set FALSE in an end station, indicates to the neighboring Bridge that the CN-TAG is to be removed from all frames, including CNMs, transmitted to the end station. It is not recommended that an end station set cnpdAcceptsCnTag to FALSE if this would result in the end station discarding CNMs for lack of a CN-TAG.

The following condition shall not cause a received CNM PDU to be considered invalid:

c) There are non-0 bits in the Version (33.4.1) or ReservedV (33.4.2).

NOTE 2—These rules permit information to be added to the CNM PDU following the Encapsulated MSDU field (33.4.10) in later revisions of this standard.

33.5 Congestion Notification TLV

The TLV illustrated in Figure 33-1 is encoded into each IEEE Std. 802.1AB LLDP message transmitted by a system that is configured to support congestion notification on one or more priority values.

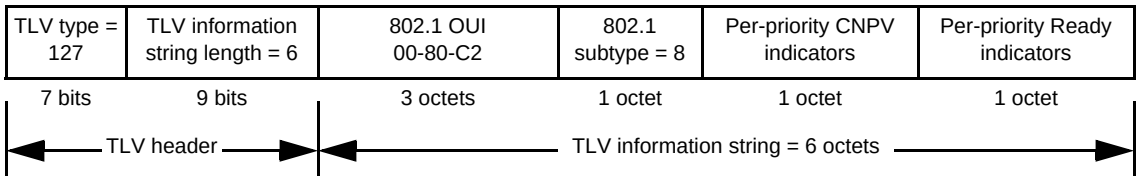

Figure 33-1—Congestion Notification TLV format

A system shall transmit no more than one Congestion Notification TLV in a single LLDP PDU. If a system receives an LLDP PDU with more than one Congestion Notification TLV, the behavior of the receiving system is unspecified. A system shall not transmit a Congestion Notification TLV with all of the Per-priority CNPV indicators (33.5.3) clear (0).

33.5.1 TLV type

A 7-bit integer value occupying the most-significant bits of the first octet of the TLV. Always contains the value 127.

33.5.2 TLV information string length

A 9-bit unsigned integer, occupying the least-significant bit of the first octet of the TLV (the most-significant bit of the TLV information string length) and the entire second octet of the TLV, containing the total number of octets in the TLV information string of the Congestion Notification TLV. This does not count the TLV type and TLV information string length fields. It is equal to 6.

33.5.3 Per-priority CNPV indicators

A bit vector, one per priority value, containing all eight of the cnpdXmitCnpvCapable variables (32.4.7) for this port. The least-significant bit of the octet carries priority 0's cnpdXmitCnpvCapable variable, and the most-significant bit that of priority 7.

33.5.4 Per-priority Ready indicators

A bit vector, one per priority value, containing all eight of the cnpdXmitReady variables (32.4.8) for this port. The least-significant bit of the octet carries priority 0's cnpdXmitReady variable, and the most-significant bit that of priority 7.

Annex A

(normative)

PICS Proforma[1]

A.5 Major Capabilities

Change the following item in A.5:

Item	Feature	Status	References	Support
MIB	Does the ~~bridge~~system implementation support management operations using SMIv2 MIB modules?	MGT: O	8.12, 17	Yes [ ] No [ ]

Insert the following items at the end of A.5:

Item	Feature	Status	References	Support
CN	Is congestion notification implemented?	O	5.4.3, 30, 31, 32, 33	Yes [ ] No [ ]
BRG-1	Is this system a bridge, and not an end station, for the purposes of congestion notification?	CN: O		Yes [ ] No [ ]

A.14 Bridge Management

Insert the following points at the end of A.14, renumbering items if necessary:

Item	Feature	Status	References	Support
MGT-130	Does the Bridge support all of the objects in the CN component managed object?	BRG-1 AND CN: M	5.4.3:c, 12.17.1	Yes [ ]
MGT-131	Does the Bridge support all of the objects in the CN component priority managed object?	BRG-1 AND CN: M	5.4.3:c, 12.17.2	Yes [ ]
MGT-132	Does the Bridge support all of the objects in the CN Port priority managed object?	BRG-1 AND CN: M	5.4.3:c, 12.17.3	Yes [ ]
MGT-133	Does the Bridge support all of the objects in the Congestion Point managed object?	BRG-1 AND CN: M	5.4.3:c, 12.17.4	Yes [ ]
MGT-134	Does the end station support all of the objects required to manage its RP(s)?	¬BRG-1 AND CN: M	5.10:d, 12.17.1, 12.17.6	Yes [ ]
MGT-135	Does the end station use the same Reaction Point group managed object to manage RPs serving more than one CNPV?	¬BRG-1 AND CN: M	32.11	No [ ]

1. *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.

Item	Feature	Status	References	Support
MGT-136	Does the end station support all of the objects in the CN component managed object?	CN-32: M	5.10:b, 12.17.1	Yes [ ]
MGT-137	Does the end station support all of the objects in the CN component priority managed object?	CN-32: M	5.10:b, 12.17.2	Yes [ ]
MGT-138	Does the end station support all of the objects in the CN Port priority managed object?	CN-32: M	5.10:b, 12.17.3	Yes [ ]
MGT-139	Does the end station support all of the objects in the Congestion Point managed object?	CN-32: M	5.10:b, 12.17.4	Yes [ ]

A.24 Management Information Base (MIB)

Insert the following items at the end of A.24:

Item	Feature	Status	References	Support
MIB-21	Does the system implement the IEEE8021-CN-MIB?	MIB AND CN: O	5.4.3:j, 5.10:n	Yes [ ] No [ ]
MIB-22	Does the system implement the System Group and Interfaces Group of the SNMPv2-MIB?	MIB-21: M	17.3.16.1	Yes [ ]
MIB-23	Does the system implement the clarifications of the Interfaces group supplied in the descriptions of ieee8021CnEpIfIndex, ieee8021CnPortPriIfIndex, ieee8021CnCpIfIndex, and ieee8021CnCpidToIfIfIndex?	MIB-21: M	17.3.16.1, 17.7.16	Yes [ ]
MIB-24	Does the system assign the same conceptual row in the IF-MIB to both an aggregated port and to one of the ports being aggregated?	MIB-21: M	17.3.16.1:b	No [ ]
MIB-25	Does the system assign a conceptual row in the IF-MIB to the individual aggregated?	MIB-21: O	17.3.16.1	Yes [ ] No [ ]
MIB-26	Does the system allow more than seven rows to be created in the ieee8021CnCompntPriTable?	MIB-21: M	17.7.16	No [ ]
MIB-27	Is the minimum value for ieee8021CnRpgMaxRate supported by the system at least equal to the required value?	MIB-21: M	17.7.16, 32.11.5	Yes [ ]
MIB-28	Is the minimum value for ieee8021CnRpgMinRate supported by the system at least equal to the required value?	MIB-21: M	17.7.16, 32.11.10	Yes [ ]

A.25 Congestion notification

Insert the following annex subclause, numbered appropriately:

Item	Feature	Status	References	Support
CN-1	Does the system conform to the required provisions of IEEE 802.1AB-2005?	CN: M	5.4.3:e, 5.10:f	Yes [ ]
CN-2	Does the system support the use of the Congestion Notification TLV in LLDP?	CN: M	5.4.3:f, 5.10:g	Yes [ ]
CN-3	Does the system transmit more than one Congestion Notification TLV in a single LLDPDU?	CN: M	33.5	No [ ]
CN-4	Does the system transmit a Congestion Notification TLV with 0 in all of the Per-priority CNPV indicators?	CN: M	33.5	No [ ]
CN-5	Does the system implement the Congestion Notification Domain defense variables, procedures, and state machine?	CN: M	32.4, 32.5, 32.6	Yes [ ]
CN-6	Does the Bridge support the creation of at least one CP on at least one Port?	BRG-1 AND CN: M	5.4.3:a	Yes [ ]
CN-7	Does the Bridge support the creation of more than one CP on at least one Port?	BRG-1 AND CN: O	5.4.3:i	Yes [ ] No [ ]
CN-8	Does the Bridge support the creation of more than seven CPs on any Port?	BRG-1 AND CN: M	5.4.3:i	No [ ]
CN-9	Does every CP on the bridge support all four defense modes separately on each CNPV?	BRG-1 AND CN: M	5.4.3:d, 31.1.1, 32.1.1	Yes [ ]
CN-10	Is each CP on the Bridge able to remove CN-TAGs?	BRG-1 AND CN: M	5.4.3:b	Yes [ ]
CN-11	Does the PIP perform CNM translation on the return path?	BEB-I AND CN:M	5.4.3:g	Yes [ ]
CN-12	Does the Provider Edge Port perform CNM translation on the return path?	PEB AND CN:M	5.4.3:h	Yes [ ]
CN-13	Is each CP on the system able to generate CNMs?	CN AND (BRG-1 OR CN-32): M	5.4.3:b, 5.10:c	Yes [ ]
CN-14	Does the bridge override the priority of a frame entering a port on a CNPV when in mode cptEdge?	BRG-1 AND CN: M	32.1.1	Yes [ ]
CN-15	Does the bridge allow any other priority to be remapped to a CNPV when in any mode other than cptDisabled?	BRG-1 AND CN: M	32.1.1	No [ ]
CN-16	Do the system's CPs implement the specified variables and procedures?	CN AND (BRG-1 OR CN-32): M	32.7:a, 32.8, 32.9	Yes [ ]
CN-17	Is the CP's Random() function initialized to a different value each time the system is reset?	CN AND (BRG-1 OR CN-32): M	32.9.1	Yes [ ]
CN-18	Does a system's CP interpret a CN-TAG to any degree beyond simply copying it to the CNM?	CN AND (BRG-1 OR CN-32): M	33.2.1	No [ ]
CN-19	Does the CP transmit a 0 in the CNM's Version field?	CN AND (BRG-1 OR CN-32): M	33.4.1	Yes [ ]

Item	Feature	Status	References	Support
CN-20	Does the CP transmit a 0 in the CNM's ReservedV field?	CN AND (BRG-1 OR CN-32): M	33.4.2	Yes [ ]
CN-21	Does the end station support the creation of at least one RP?	¬BRG-1 AND CN: M	5.10:a	Yes [ ]
CN-22	Does the end station allow more than rpppMaxRps RPs to be created on one port and priority?	¬BRG-1 AND CN: M	32.10.1	No [ ]
CN-23	Does the end station support at least the cptDisabled and cptInterior defense modes separately on each CNPV?	¬BRG-1 AND CN: M	5.10:e, 32.1.1	Yes [ ]
CN-24	Does the end station add a tag to frames transmitted from CCFs?	¬BRG-1 AND CN: O	30.5, 31.2.2.5	Yes [ ] No [ ]
CN-25	Does the end station support both the cptInterior, and cptInteriorReady defense modes separately on each CNPV?	¬BRG-1 AND CN: O	5.10:m, 31.2.2.5, 32.14.3	Yes [ ] No [ ]
CN-26	Does the end station support the cptEdge defense mode on any CNPV?	¬BRG-1 AND CN: O	5.10:e, 32.4.9	Yes [ ] No [ ]
CN-27	Does the end station accept CN-TAGs in data frames on any CNPV?	¬BRG-1 AND CN: O	5.10:e, 32.4.10	Yes [ ] No [ ]
CN-28	Does the end station support the creation of more than one RP?	¬BRG-1 AND CN: O	5.10:k, 31.2.1	Yes [ ] No [ ]
CN-29	Does the end station support the creation of more than one RP?	¬BRG-1 AND CN: O	5.10:l, 31.2.1	Yes [ ] No [ ]
CN-30	Do the end station's RP(s) limit the output frame rate in response to CNMs received?	¬BRG-1 AND CN: M	5.10:h	Yes [ ]
CN-31	Does the end station distinguish correctly to which RP a given CNM is directed?	CN-28: M	5.10:i	Yes [ ]
CN-32	Does the end station support the creation of at least one CP?	¬BRG-1 AND CN: O	5.10:j	Yes [ ] No [ ]
CN-33	Does the end station's have a single flow queue as its default configuration?	¬BRG-1 AND CN: M	31.2.1	Yes [ ]
CN-34	Does the end station change its method for assigning frames to Flow queues in a manner that impairs the ability of compliant CPs to throttle its flows?	¬BRG-1 AND CN: M	31.2.1	No [ ]
CN-35	Does the end station change its method for assigning frames to Flow queues in a manner that significantly increases the likelihood of frame misordering, and thus impacts higher layer functions?	¬BRG-1 AND CN: M	31.2.1	No [ ]
CN-36	Does the end station's Rate Limiters always meet the specified formula for L_T?	¬BRG-1 AND CN: M	31.2.2.4 equation (1)	Yes [ ]
CN-37	Can the end station's Flow Selection functions drop frames between the Flow queue and the Flow multiplexer?	¬BRG-1 AND CN: M	31.2.2.5	No [ ]
CN-38	Does the end station output frames with CN-TAGs on a priority operating in mode cptEdge or cptInterior?	¬BRG-1 AND CN: M	32.1.1	No [ ]
CN-39	Do the end station's RPs implement the specified timers, variables. procedures, and state machines?	¬BRG-1 AND CN: M	32.7:b, 32.12, 32.13, 32.14, 32.15	Yes [ ]

Item	Feature	Status	References	Support
CN-40	Does the end station insert the CN-TAG and VLAN-tag in the correct order (addresses, VLAN-tag, CN-TAG)?	¬BRG-1 AND CN: M	33.2	Yes [ ]
CN-41	Does the end station ignore the CNM's Version field?	¬BRG-1 AND CN: M	33.4.1	Yes [ ]
CN-42	Does the end station ignore the CNM's ReservedV field?	¬BRG-1 AND CN: M	33.4.2	Yes [ ]
CN-43	Does the end station assign meaning to subfields within a received CNM's Congestion Point Identifier field?	¬BRG-1 AND CN: M	33.4.4	No [ ]
CN-44	Does the end station ignore received CNMs that are too short?	¬BRG-1 AND CN: M	33.4.11:a	Yes [ ]
CN-45	Does the end station ignore received CNMs that have no CN-TAG?	¬BRG-1 AND CN: O	33.4.11:b	Yes [ ] No [ ]
CN-46	Does the end station ignore received CNMs that have non-0 values in the Version or ReservedV fields?	¬BRG-1 AND CN: M	33.4.11:c	No [ ]

Annex H

(informative)

Bibliography

Change the following bibliographic citations, and arrange in numerical order, renumbering if necessary:

~~[B26] IEEE Std 802.1AB-2005, IEEE Standard for Local and Metropolitan Area Networks—Station and Media Access Control Connectivity Discovery.~~

[B40] IETF RFC 793 (STD0007), Transmission Control Protocol.

[B41] IETF RFC 1323 (Proposed Standard), TCP Extensions for High Performance.

[B42] IETF RFC 2581 (Proposed Standard), TCP Congestion Control.

[B43] IETF RFC 2960 (Proposed Standard), Stream Control Transmission Protocol.

[B44] M. Alizadeh, B. Atikoglu, A. Kabbani, A. Lakshmikantha, R. Pan, B. Prabhakar and M. Seaman; "Data Center Transport Mechanisms: Congestion Control Theory and IEEE Standardization"; Proceedings of The 46th Annual Allerton Conference on Communication, Control and Computing; Urbana-Champaign, Sept. 2008; Invited paper.

[B45] Binary Increase Congestion control - Transmission Control Protocol (BIC-TCP); http://netsrv.csc.ncsu.edu/twiki/bin/view/Main/BIC.