8. Principles of bridge operation

<<Editor's Note: The main substantive changes are going to be in Clause 8, and particularly in the transmission selection subclauses. I have also included the "queuing frames" subclause as there may be a need to revise the mapping recommendations, but so far I haven't specified any changes there.>>

8.6 The Forwarding Process

8.6.6 Queuing frames

Change the text of 8.6.6 as shown:

The Forwarding Process provides storage for queued frames, awaiting an opportunity to submit these for transmission. The order of frames received on the same Bridge Port shall be preserved for

a) unicast frames with a given VID, priority, and destination address and source address combination;
b) multicast frames with a given VID, priority, and destination address.

The Forwarding Process provides one or more queues for a given Bridge Port, each corresponding to a distinct traffic class. Each frame is mapped to a traffic class using the Traffic Class Table for the Port and the frame's priority. Traffic class tables may be managed. Table 8-4 shows the recommended mapping for the number of classes implemented, in implementations that do not support the credit-based shaper transmission selection algorithm (8.6.8.2). The requirements for priority to traffic class mappings in implementations that support the credit-based shaper transmission selection algorithm are defined in 34.5. Up to eight traffic classes may be supported, allowing separate queues for each priority.

Table 8-4—Recommended priority to traffic class mappings

		Number of available traffic classes							
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Priority	0 (Default)	0	0	0	0	0	1	1	1
	1	0	0	0	0	0	0	0	0
	2	0	0	0	1	1	2	2	2
	3	0	0	0	1	1	2	3	3
	4	0	1	1	2	2	3	4	4
	5	0	1	1	2	2	3	4	5
	6	0	1	2	3	3	4	5	6
	7	0	1	2	3	4	5	6	7

NOTE—The rationale for these mappings is discussed in Annex I.

NOTE 1—Different numbers of traffic classes may be implemented for different Ports. Ports with media access methods that support a single transmission priority, such as CSMA/CD, can support more than one traffic class.

NOTE 2—A queue in this context is not necessarily a single FIFO data structure. A queue is a record of all frames of a given traffic class awaiting transmission on a given Bridge Port. The structure of this record is not specified. The

transmission selection algorithm (8.6.8) determines which traffic class, among those classes with frames available for transmission, provides the next frame for transmission. The method of determining which frame within a traffic class is the next available frame is not specified beyond conforming to the frame ordering requirements of this subclause. This allows a variety of queue structures such as a single FIFO, or a set of FIFOs with one for each pairing of ingress and egress ports (i.e., Virtual Output Queuing), or a set of FIFOs with one for each VLAN or priority, or hierarchical structures.

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.

Where the peristaltic shaper algorithm has been assigned as the transmission selection algorithm for a given traffic class (8.6.8), each frame that is queued for that traffic class is assigned a *peristaltic phase label* that carries the value of the *peristaltic reception phase* (8.6.8.5) for that queue at the time the frame was queued.

8.6.8 Transmission selection

Change the text of 8.6.8 and its subclauses as shown:

For each Port, frames are selected for transmission on the basis of the traffic classes that the Port supports, ~~and~~ the operation of the transmission selection algorithms supported by the corresponding queues on that Port, and the state of the transmission gates (8.6.8.4) associated with those queues. For a given Port and supported value of traffic class, frames are selected from the corresponding queue for transmission if and only if

a) The operation of the transmission selection algorithm supported by that queue determines that there is a frame available for transmission; and

b) The transmission gate (8.6.8.4) associated with that queue is in the *open* state; and

c) If there are gate events associated with that queue, then there is sufficient time available to transmit the entireity of that frame before the next gate-close event associated with that queue; and

d) For each queue corresponding to a numerically higher value of traffic class supported by the Port, the operation of the transmission selection algorithm supported by that queue determines that there is no frame available for transmission.

In a port of a Bridge or station that supports PFC, a frame of priority n is not available for transmission if that priority is paused (i.e., if Priority_Paused[n] is TRUE (see 36.1.3.2)) on that port. When Transmission Selection is running above Link Aggregation, a frame of priority n is not available for transmission if that priority is paused on the physical port to which the frame is to be distributed.

NOTE 1—Two or more priorities can be combined in a single queue. In this case if one or more of the priorities in the queue are paused, it is possible for frames in that queue not belonging to the paused priority to not be scheduled for transmission.

NOTE 2—Mixing PFC and non-PFC priorities in the same queue results in non-PFC traffic being paused causing congestion spreading, and therefore is not recommended.

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

The Transmission Selection Algorithm Table for a given Port assigns, for each traffic class that the Port supports, the transmission selection algorithm that is to be used to select frames for transmission from the corresponding queue. Transmission Selection Algorithm Tables may be managed, and allow the identification of vendor-specific transmission selection algorithms. The transmission selection algorithms are identified in the Transmission Selection Algorithm Table by means of integer identifiers, as defined in Table 8-5.

Table 8-5—Transmission selection algorithm identifiers

Transmission selection algorithm	Identifier
Strict priority (8.6.8.1)	0
Credit-based shaper (8.6.8.2)	1
Enhanced Transmission Selection (8.6.8.3)	2
Peristaltic shaper (8.6.8.2)	3
Reserved for future standardization	~~3~~4-254
Vendor-specific Transmission Selection algorithm value for use with DCBX (Clause D.2.9.8)	255
Vendor-specific	A four-octet integer, where the 3 most significant octets hold an OUI value, and the least significant octet holds an integer value in the range 0–255 assigned by the owner of the OUI.

In implementations that do not support the credit-based shaper transmission selection algorithm, the recommended default configuration for the Transmission Selection Algorithm Tables is to assign the strict priority transmission selection algorithm to all supported traffic classes. In implementations that support the credit-based shaper transmission selection algorithm, the recommended default configuration for the Transmission Selection Algorithm Tables is defined in 34.5.

8.6.8.1 Strict priority algorithm

For a given queue that supports strict priority transmission selection, the algorithm determines that there is a frame available for transmission if the queue contains one or more frames. The order in which frames are selected for transmission from the queue shall maintain the ordering requirement specified in 8.6.6.

8.6.8.2 Credit-based shaper algorithm

For a given queue that supports credit-based shaper transmission selection, the algorithm determines that a frame is available for transmission if the following conditions are all true:

a) The queue contains one or more frames.
b) The *transmitAllowed* signal for the queue is TRUE.

The order in which frames are selected for transmission from the queue shall maintain the ordering requirement specified in 8.6.6.

The following external parameters are associated with each queue that supports the operation of the credit-based shaper algorithm:

c) ***portTransmitRate.*** The transmission rate, in bits per second, that the underlying MAC service that supports transmission through the Port provides. The value of this parameter is determined by the operation of the MAC.

d) ***idleSlope*****.** The rate of change of *credit*, in bits per second, when the value of *credit* is increasing (i.e., while *transmit* is FALSE). The value of *idleSlope* can never exceed *portTransmitRate*. The value of *idleSlope* for a given queue is determined by the value of the ***operIdleSlope(N)*** parameter for that queue, as defined in 34.3.

The following internal parameters are associated with each queue that supports the credit-based shaper algorithm:

e) ***transmit***. Takes the value TRUE for the duration of a frame transmission from the queue; FALSE when any frame transmission from the queue has completed.

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

g) ***sendSlope*****.** The rate of change of *credit*, in bits per second, when the value of *credit* is decreasing (i.e., while *transmit* is TRUE). The value of *sendSlope* is defined as follows:

$$sendSlope = (idleSlope - portTransmitRate)$$

h) ***transmitAllowed.*** Takes the value TRUE when the *credit* parameter is zero or positive; FALSE when the *credit* parameter is negative.

NOTE 1—The value of *credit* is naturally constrained by the operating parameters for the algorithm, and also by the operation of any higher priority queues that use this algorithm. The lowest value that *credit* can reach depends on *sendSlope* and the largest frame that will be transmitted from the queue. This largest frame size is a consequence of the type of traffic that is being transmitted using the queue, rather than a limit imposed by the algorithm or by management. The highest value that can be accumulated in *credit* depends on *idleSlope* and the length of time that the algorithm may have to wait when the queue is not empty and there is other traffic being transmitted through the Port; for any given queue, that length of time is bounded, as discussed in Annex L, which also illustrates how the algorithm operates under varying conditions, and how its operation affects the maximum latency that can be experienced by SR traffic.

NOTE 2—In order for the credit-based shaper algorithm to operate as intended, it is necessary for all traffic classes that support the algorithm to be numerically higher than any traffic classes that support the strict priority algorithm defined in 8.6.8.1. The mapping of priorities onto traffic classes, and the choice of traffic classes that support particular transmission selection algorithms, is defined in 34.5.

Traffic classes using the credit-based shaper algorithm shall not use PFC and shall ignore the setting of the bits related to such classes in the PFC Enable bit vector (see 38.5.4.6 in IEEE Std 802.1Qaz-2011)

8.6.8.3 Enhanced Transmission Selection algorithm

If ETS is enabled for a traffic class, transmission selection is performed based on the allocation of bandwidth to that traffic class. Bandwidth is distributed amongst ETS traffic classes that support enhanced transmission selection algorithm such that each traffic class is allocated available bandwidth in proportion to its TCBandwidth (see Clause 37).

For a given queue that supports enhanced transmission selection, the algorithm determines that there is a frame available for transmission if the following conditions are all true:

a) The queue contains one or more frames;

b) The ETS algorithm (37.3) determines that a frame should be transmitted from the queue; and

c) There are no frames available for transmission for any queues running Strict priority or Credit based shaper algorithms.

The order in which frames are selected for transmission from the queue shall maintain the ordering requirement specified in 8.6.6.

Insert new subclause 8.6.8.4 and Table 8-6 as shown, renumbering subsequent tables as necessary:

8.6.8.4 Transmission gates, gate states, and gate events

A transmission gate is associated with each queue; the state of the transmission gate determines whether or not queued frames can be selected for transmission. For a given queue, the transmission gate can be in one of two states:

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

An implementation may support a list of gate events associated with each queue that change the transmission gate state for that queue. In an implementation that does not support gate event lists, all gates are permanently in the *open* state. Table 8-6 identifies the gate event types, their parameters, and the actions that result from their execution.

Table 8-6—Gate events

Event name	Event parameter	Event action
Gate-close	time-interval	Change the gate state to *closed* after *time-interval* has elapsed, i.e., causes a gate-closing event (3.2) to occur.
Gate-open	time-interval	Change the gate state to *open* after *time-interval* has elapsed, i.e., causes a gate-opening event (3.3) to occur.
Repeat	time-interval	Repeat the list of events, from the first event in the list, after *time-interval* has elapsed.

Execution of the gate event list starts on system initialization, or after any changes are made to the event list, with the first gate event in the list. A time-interval associated with a gate event is measured relative to the time at which the previous gate event in the list completed its execution. The time at which the first gate event in the gate event list starts execution, *tFirstGateEvent*, is determined as follows:

$$\text{tFirstGateEvent} = n \times \text{tCycle}$$

where n is an integer, and tCycle is the length of the gating cycle (3.4) for the queue, calculated as the sum of the time-interval parameters of the events in the list up to and including the Repeat event. tFirstGate event is expressed in terms of the PTP timescale; it is the next value of PTP time after system startup that is an integer multiple of tCycle.

All time-interval parameter values are specified with a granularity of 1 nanosecond, and specify a time interval relative to the time at which the previous gate event completed execution.

<<Editor's Note: I have amended the text, following the discussion in San Diego, to work on the basis of PTP time rather than specifying a start time in the event list itself.>>

Insert new subclause 8.6.8.5 as shown:

8.6.8.5 Peristaltic shaper algorithm

For a given queue that supports peristaltic shaper transmission selection, the algorithm determines that a frame is available for transmission if the following condition is true:

a) The queue contains one or more frames whose *peristaltic phase label* (8.6.6) matches the current *peristaltic transmission phase* for that queue.

The order in which frames are selected for transmission from the queue shall maintain the ordering requirement specified in 8.6.6.

The following internal parameters are associated with each queue that supports the peristaltic shaper algorithm:

b) *Peristaltic reception phase (pRphase)*: The phase value to be assigned to frames on reception. This parameter takes the value *even* or *odd*, and is determined as described below.

c) *Peristaltic transmission phase (pTphase)*: The current peristaltic transmission phase. This parameter takes the value *even* if *pRphase* is *odd*, or *odd* if *pRphase* is *even*.

d) *Peristaltic cycle time (pCycle)*: The period of time, in nanoseconds, between changes in the value of the *pRphase.*

The value of *pRphase* is derived from the current time known to the system. If:

INT (current_PTP_time / pCycle)

is an odd number, then *pRphase* takes the value *odd*; otherwise, *pRphase* takes the value *even.*

where current_PTP_time is expressed in terms of the PTP timescale; it is the value of PTP time that represents the current time.

NOTE—The intent is that pCycle is scaled according to the speed of transmission of the port and the length of time needed to accommodate the maximum burst of traffic that will use the shaper, plus the maximum sized interfering frame or frame fragment.

<<Editor's Note: I have chosen to describe the peristaltic shaper in terms of labels and transmission phases using a single queue; however, it will be apparent that there is an exaclty equivalent description in which two queues are used, and the transmission phases are achieved simply by making use of transmission gates and gate events. However, I felt that the latter description might end up lacking clarity & involving further complexities, due to the fact that this would be a first situation in which a single traffic class feeds two distinct outbound queues. I prefer the clarity of the description as presented; however, the reality is that the 2 queues version is directly interchangeable from an implementation standpoint.>>