Text Proposal for:
A (sub-)section on Shaper Interoperability in IEEE 802.1DG

Max Turner
Ethernovia
(BMW, Ruetz System Solutions)
June 2021, v5

1 Glossary

1.1 General note on nomenclature

This paper is looking at bringing together historical versions of IEEE Std. 802.1Q along with its amendments, representing the transition from AVB to TSN, as well as IEEE Std. 802.1BA which has not been updated since its publication in 2011. This inevitably leads to imperfections in nomenclature. An effort was made to try use a consistent terminology within the paper without loosing the linkages with the IEEE documents. The reader is advised to exercise caution when comparing and searching for definitions.

1.2 Latency

The time it takes for a full frame to pass from a transmitter to a receiver. It is often referred to as the time “first bit out to last bit in”.

1.3 Delay

The time it takes for a particular element of a frame (usually the start of the fame) to pass from a transmitter to a receiver. It is often referred to as the time “first bit out to first bit in”.

1.4 TAS – Time Aware Shaper

Originally specified in IEEE Std. 802.1Qbv Enhancements to Traffic Scheduling.

1.5 CBS – Credit Based Shaper

Originally specified in IEEE Std. 802.1Qav as Forwarding and Queuing Enhancements for Time-Sensitive Streams (FQTSS).

1.6 ATS – Asynchronous Shaper

Originally specified in IEEE Std. 802.1Qcr Asynchronous Traffic Shaping.

1.7 AVB Profile

IEEE Std. 802.1BA was the initial Audio/Video profile in for Audio/Video-Bridging (AVB), the precursor of Time-Aware-Shaping (TSN) described in IEEE Std. 802.1Q and it’s various amendments.

individual contribution, to be presented 2021-06-29 – updated since 2021-06-01 with input from inside and outside the working group

1.8 portTransmitRate

Refers to the line rate of a port. Any frame is always transmitted as a burst using portTransmitRate (optionally abbreviated pTR in equations).

1.9 operCycleTime (TAS)

Is the time after which the list of TAS gate control entries repeats.

1.10 gateOpenTime (TAS)

Is the total time a particular TAS gate under consideration is open during operCycleTime. This may be summarizing the times of multiple gate control entries and is therefor not necessarily continuous as per IEEE Std. 802.1Q. This is further discussed in section 5.2.1.

1.11 tasBand (TAS)

Is the average bandwidth available to a queue controlled by a TAS gate, which is open for operCycleTime during gateOpenTime:

tasBand = portTransmitRate * gateOpenTime / operCycleTime .... (1.11-1)

Therefore:

tasBand ≤ portTransmitRate .... (1.11-2)

This does not take into account limited usability due to frame size.

1.12 maxAllocBand (CBS)

In IEEE Std. 802.1BA this represents the maximum average bandwidth allowed to be allocated for CBS. As IEEE Std. 802.1BA did not consider the interaction between TAS and CBS this paper will discuss whether it should be viewed as the maximum value for idleSlope or for idleSlope (main part of section 5.3 and section 5.3.2).

IEEE Std. 802.1Q gives such a limit, which is used in IEEE Std. 802.1BA. The rationale is further discussed in section 5.3.1.

1.13 cbsBits (CBS)

Is the number of bits which have been allocated to be transmitted by CBS during the observationInterval given by the streams SR Class.

1.14 SR Class (CBS)

Defines the observationInterval, also referred to as class measurement interval in IEEE Std. 802.1Q, to be used in bandwidth calculations for the CBS.

1.15 tAllStreams (CBS)

Within one SR Class frames from multiple streams may be transmitted during the observationInterval of the CBS. The actual time needed to transmit all of these at portTransmitRate is denoted by tAllStreams.

tAllStreams = cbsBits / portTransmitRate .... (1.15-1)

In IEEE Std. 802.1BA only the maximum allocation and usage is considered. This paper allows for overprovisioning.

1.16 observationInterval (CBS)

Is used as the time base to calculate allocated bandwidth and therefore credit aggregation for CBS.

The SR Classes define default observationInterval values mostly derived from audio transport use cases on 100Mbit/s links. This paper does not limit the discussion to observatioInterval values defined by SR Classes.

From IEEE Std. 802.1Q:

> “The sole implication of the observation interval is its effect on frame size, because the shaper behavior itself is independent of the observation interval. The intent in limiting the observation interval is to limit frame size, as this is a major contribution to the latency experienced by a frame in transit through an audio/video bridging (AVB) network”

1.17 idleSlope (CBS)

Is used to express the rate at which credit is accumulated for the CBS if a TAS gate is enabled. It can be calculated from:

idleSlope = operIdleSlope * operCycleTime / gateOpenTime .... (1.17-1)

If no TAS gate is enabled this is identical to the bandwidth allocated for CBS (operIdleSlope). In any case:

idleSlope ≥ operIdleSlope .... (1.17-2)

Equation (1.17-1) is derived in section 5.1 of this paper and also given in IEEE Std. 802.1Q.

Since idleSlope ≤ portTransmitRate, (1.17-1) can be used to require:

portTransmitRate ≥ (operCycleTime * operIdleSlope) / gateOpenTime .... (1.17-3)

1.18 operIdleSlope (CBS)

Is the bandwidth used by CBS. It can be calculated from the number of bits which need to be transferred during one observationInterval determined by the SR Class:

operIdleSlope = cbsBits / observationInterval .... (1.18-1)

If no TAS gate is enabled this is identical to the rate at which credit is accumulated for the CBS.

The description of CBS in IEEE Std. 802.1Q and IEEE Std. 802.1BA assumes for the allocation of bandwidth to be equal to the bandwidth currently used. Combing (1.15-1) and (1.18-1) then yields:

operIdleSlope = (tAllStreams / observationInterval) * portTransmitRate .... (1.18-2)

This is intended to be ensured by using e.g. SRP to adapt the allocation of bandwidth dynamically to the bandwidth currently used.

1.19 CBS overprovisioning

Early on in the development in AVB it was discussed, that latency for a certain stream could be reduced by overprovisioning the allocated bandwidth, meaning:

operIdleSlope > (tAllStreams / observationInterval) * portTransmitRate .... (1.19-1)

This was also found to be a useful concept to avoid running SRP and setting the allocation to a maximum in all bridges.

In this case (1.18-1) no longer holds true! Instead operIdleSlope becomes a configuration parameter which is then limited by maxAllocBand.

2 IEEE802.1Q Egress Model

Within the transmission selection model of IEEE802.1Q, it is possible to logically couple CBS, TAS, and SP in series. This means in order for a frame to be transmitted, all three elements must allow the transmission at the exact same time and a frame must be available for transmission.

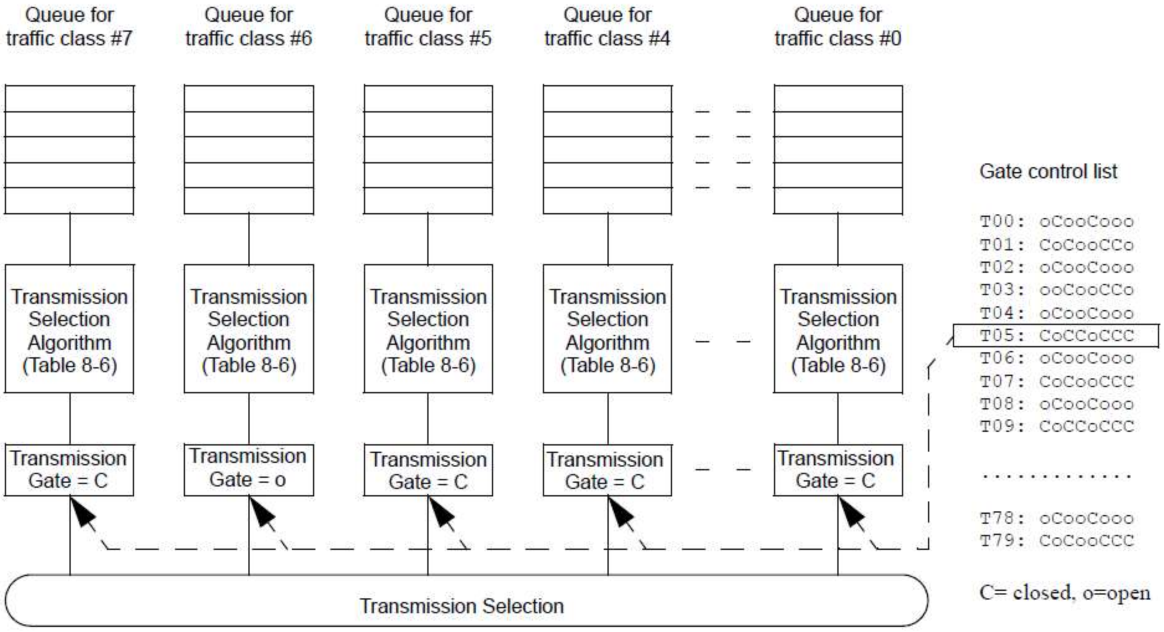
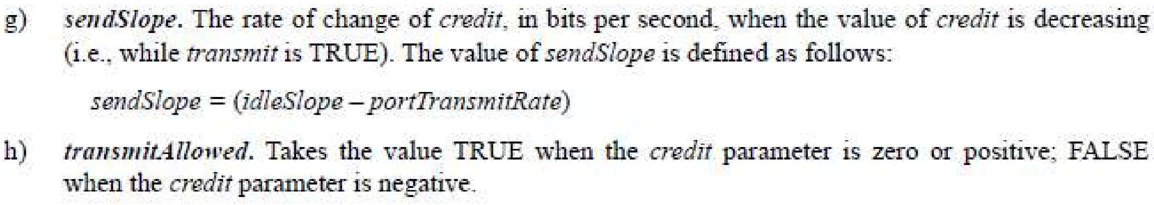

Queue for traffic class #7
Queue for traffic class #6
Queue for traffic class #5
Queue for traffic class #4
Queue for traffic class #0
Transmission Selection Algorithm (Table 8-6)
Transmission Selection Algorithm (Table 8-6)
Transmission Selection Algorithm (Table 8-6)
Transmission Selection Algorithm (Table 8-6)
Transmission Selection Algorithm (Table 8-6)
Transmission Gate = C
Transmission Gate = o
Transmission Gate = C
Transmission Gate = C
Transmission Gate = C
Transmission Selection
Gate control list
T00: oCooCooo
T01: CoCooCCo
T02: oCooCooo
T03: ooCooCCo
T04: oCooCooo
T05: CoCCoCCC
T06: oCooCooo
T07: CoCooCCC
T08: oCooCooo
T09: CoCCoCCC
....
T78: oCooCooo
T79: CoCooCCC
C= closed, o=open

Figure 8-14—Transmission selection with gates

The TAS gate is completely unaware of the state of the queue (is a frame available?). It is controlled by a clock cycle, where two basic patterns of control are discussed below.

The CBS has a (dynamically) configured bandwidth expressed as the operIdleSlope, which accumulates credit (idleSlope) only if a frame is in the queue and the TAS gate is open.



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.

The ATS determines a transmission time based on the arrival time of the frame, potentially using a different time base than the TAS.

3 Time Aware Shaper (TAS)

3.1 TAS Bus Mode

In what this document refers to as the Bus Mode, the gate control of all ports on a LAN are controlled by a single Gate Control List, whose Gate Open Times and Gate Open Duration are identical. This means the whole LAN is reserved for the transmission of a single set of messages. It is called Bus Mode as it resembled the operation of a Bus systems with a TDMA schema. In Bus Mode care must be taken to choose the Gate Open Duration long enough for the frame to propagate the maximum number of hops required. The Gate Open Duration must be at least the Latency of the set of messages to be transferred. This will usually mean having to increase the Gate Open Duration when a hop is added to the network and thereby potentially changing the whole TAS Gate Schedule in each hop of the LAN. Clock synchronisation inaccuracies across all hops need to be taken into account when determining the schedule. As a rough approximation the Gate Open Duration will be at least the number of hops times the length of the set of messages to be transported. For 100Mbit/s link speeds this will likely be in the 100’s of µs, for 1Gbit/s link speeds this will likely be in the 10’s of µs. These numbers are given to give an indication on the required resolution of the required time synchronization.

The number of such protected gate times is assumed to be limited as the whole LAN is blocked for the duration. If Bus Mode is allowed, the Gate Control List shall have at least 8 entries with a resolution of at least 10µs.

While the fames in question are allowed to pass undisturbed through the LAN, it must be considered that this time is forced onto other frames as delay.

3.2 TAS Phased Mode

In what this document refers to as the Phased Mode, the gate control of each port a set of messages traverses is timed to allow the transmission without any queuing delay after reception. In a daisy chain this means the port gate of the next hop opens about the transmission latency plus the store and forward latency later than on the previous hop. The naming has been derived form phased traffic lights, which allow vehicles to get to the next intersection when the traffic lights are green. This requires exact a priory knowledge of the path the messages to be protected will travel along. It also means that any minute change in the transmission timing needs to be considered along the full path and will usually lead to a recalculation of the schedule for each hop. To increase the net usage of any egress port it will also require more accurate time resolution than Bus Mode.

The number of such protected gate times is assumed to be high compared to Bus Mode. If Phased Mode is allowed, the Gate Control List shall have at least 32 entries with a resolution of at least 64 byte times of the line rate of the port in question (64*8bit/100Mbit/s = 5µs, 64*8bit/1Gbit/s = 0.5µs).

3.3 TAS Blockage Protection

As TAS will check for the length of a frame to ensure it can be transmitted in the allocated gate open cycle, it must be ensured that no frame with a length larger than the maximum gate open time within a Gate Control List is allowed in the egress queue. As such a mismatch could constitute an unrecoverable blockage of the port, it is required to required for any node implementing TAS to also implement a policer on frame length.

Note: IEEE Std. 802.1Q section 8.6.8.4 states: “frames that exceed queueMaxSDU are discarded [item b8) in 6.5.2].”

NOTE in 802.1Q section 8.6.9.2.2: SetGateStates(): A frame is not sent when TX can't be completed before the gate closes. “NOTE—If the OperGateStates value differs from the previous gate states, the SetGateStates() procedure causes gateopen and/or gate-close events to occur (3.97, 3.98). It is possible that, on a given queue, the maximum time interval between any gate-open event and a subsequent gate-close event is smaller than the value of queueMaxSDU, in which case, frames that are too long to transmit but are shorter than queueMaxSDU could be queued on that queue. Such frames would never be transmitted, but would eventually be discarded because they exceed the maximum frame lifetime (6.5.2, 6.5.6). It should also be noted that a Bridge is allowed to discard frames that could never be transmitted [item b8) in 6.5.2].”

4 Buffer Overrun Protection

Misconfiguration of any TSN shaping algorithm may lead to a (slow) build up of residual queue occupancy. It is required for any node to implement a watchdog on queue occupancy, which is able to clear the queue (loss of data!) in case it reaches a threshold occupancy. While this will lead to the loss of data, it prevents the overall flow of data from being completely stopped.

The maximum occupancy threshold should be set at a few 10kbyte, given a regular max. Size Ethernet frame has 1.5kByte.

Further action and error recording of such an event is left to the implementer.

Note: IEEE Std. 802.1Q section 6.5.6 suggests a Maximum Bridge transit delay of 1 to 4 seconds. At 100Mbit/s line rate, this could require a buffer of up to 50Mbyte.

5 Combination of CBS and TAS

5.1 No-Interference Stability Condition

Assuming there is no interfering traffic for a CBS controlled queue because the TAS gate is open for only this one queue, there is a simple stability condition that needs to be met, in order to not overrun the buffer of the queue.

The CBS streams can deliver a maximum of (operIdleSlope * OperCycleTime) during one TAS gate cycle. While the gate is open, (portTransmitRate * GateOpenTime) can be transmitted. For stability at least as many bits must be transmitted as could be received:

$$\mathbf{(operIdleSlope * OperCycleTime) \leq (portTransmitRate * GateOpenTime)} \tag{5.1-1}$$

The CBS stream is assumed to be quantized in n Frames of equal length L:

$$\mathbf{(operIdleSlope * OperCycleTime) = (n * L)} \tag{5.1-2}$$

and one can find an:

$$\mathbf{n = \lceil (operIdleSlope * OperCycleTime) / L \rceil} \tag{5.1-3}$$

So the configuration is stable if and only if:

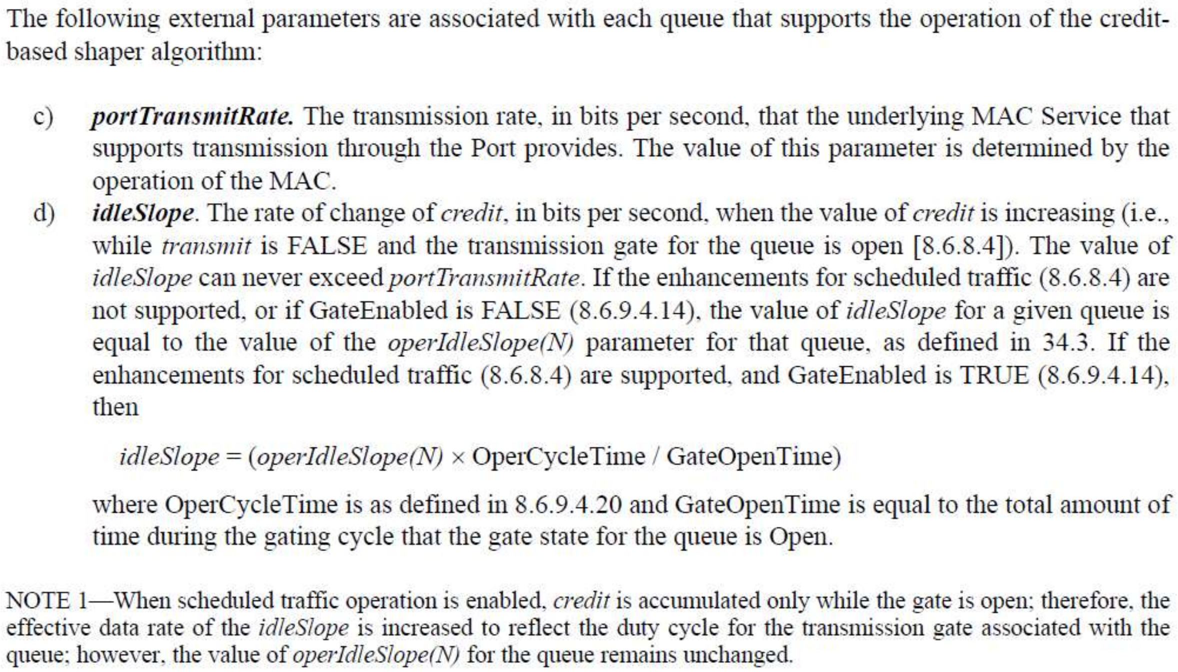
$$\mathbf{\lceil (operIdleSlope * OperCycleTime) / L \rceil * L \leq (portTransmitRate * GateOpenTime)} \tag{5.1-4}$$

Defining an average bandwidth (idleSlope) for which the same number of bits can be transmitted during GateOpenTime as will be received on average during OperCycleTime, leads to:

$$\mathbf{operIdleSlope * OperCycleTime = idleSlope * GateOpenTime} \tag{5.1-5}$$

This is identical to (1.17-1) and is also given in IEEE 802.1Q.

IEEE Std 802.1Q-2018 page 201



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 and the transmission gate for the queue is open [8.6.8.4]). The value of *idleSlope* can never exceed *portTransmitRate*. If the enhancements for scheduled traffic (8.6.8.4) are not supported, or if GateEnabled is FALSE (8.6.9.4.14), the value of *idleSlope* for a given queue is equal to the value of the *operIdleSlope(N)* parameter for that queue, as defined in 34.3. If the enhancements for scheduled traffic (8.6.8.4) are supported, and GateEnabled is TRUE (8.6.9.4.14), then

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

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

NOTE 1—When scheduled traffic operation is enabled, *credit* is accumulated only while the gate is open; therefore, the effective data rate of the *idleSlope* is increased to reflect the duty cycle for the transmission gate associated with the queue; however, the value of *operIdleSlope(N)* for the queue remains unchanged.

As per the CBS bandwidth definition (without overprovisioning from section 1.19), the reservation is done according to (1.18-1). Assuming (operIdleSlope * OperCycleTime) = cbsBits and (portTransmitRate * GateOpenTime) = cbsBits then forces:

operCycleTime = observationInterval ....(5.1-6)

Obviously this is still true if operCycleTime is an integer multiple of observationInterval, if gateOpenTime is scaled accordingly.

Using (1.17-1) and (5.1-6), one can re-write (1.15-1) into multiple identities for this case:

tAllStreams = (operIdleSlope * observationInterval) / portTransmitRate ....(5.1-7)

= (operIdleSlope * operCycleTime) / portTransmitRate ....(5.1-8)

= (idleSlope * gateOpenTime) / portTransmitRate ....(5.1-9)

Meaning the egress port must actually be busy transmitting for tAllStreams during operCycleTime and:

gateOpenTime ≥ tAllStreams..(5.1-10)

in order to achieve stability.

5.1.1 Burst Operation

For the extreme case of:

tAllStreams = GateOpenTime ....(5.1-11)

from (5.1-10) and using (1.11-1) one can rewrite (5.1-8):

operCycleTime = TASBand / (gateOpenTime / operCycleTime) * (tAllStreams / operIdleSlope)

operIdleSlope = TASBand ....(5.1-12)

Meaning the gateOpenTime from (5.1-11) allocated to the TAS results in an identical bandwidth to what is allocated to CBS by operIdleSlope. The output will be a single burst while the TAS gate is open, no other gate can be allowed to be open and no effect of the CBS will be visible to the receiver on the link.

For burst operation case the CBS shall be disabled!

5.1.2 Singular Operation

Still assuming the queue in question is the only one allowed to transmit, but the bandwidth allowed by the TAS gate timing (tasBand) is exactly the average bandwidth (idleSlope) to fulfil the stability criterion of (5.1-5), meaning:

TASBand = idleSlope ....(5.1-13)

Yieds:

pTR * (gateOpenTime / operCycleTime) = operIdleSlope * (operCycleTime / gateOpenTime)

operIdleSlope / pTR = (gateOpenTime / operCycleTime)^2

Further using (5.1-6) yields:

gateOpenTime = sqrt(operIdleSlope / portTransmitRate) * observationInterval ....(5.1-14)

Meaning a fixed configuration of the TAS, based on the CBS parameters. This limits the useability for other applications, which may require CBS independent TAS configuration.

5.2 Multi Gate Operation

Section 5.1 and its subsections make two assumptions: They do not allow any interfering traffic, as only one TAS gate is open, and they assume a single observationInterval, meaning a single SR Class for all CBS controlled queues behind a single TAS gate. Furthermore no overprovisioning according to section 1.19 is considered.

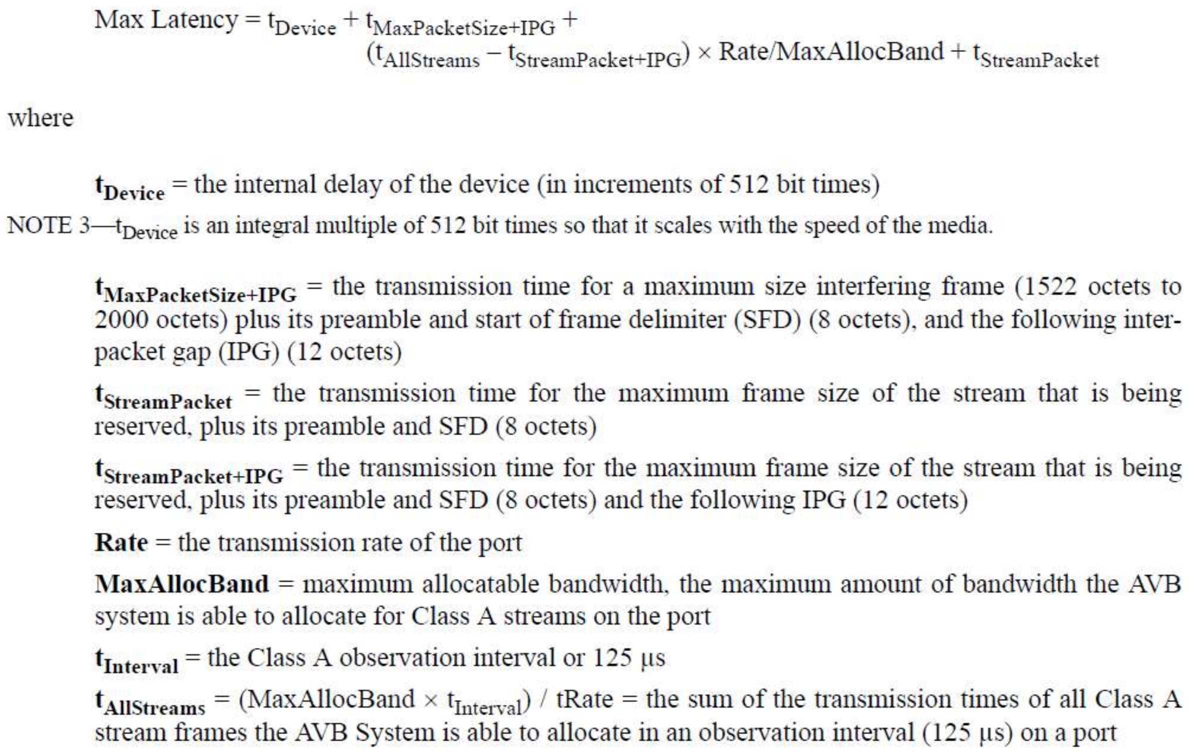
If at least one more gate is open along with the CBS and TAS controlled queue in question, it is important to configure the gate open time to allow for any interfering traffic along with the bandwidth reserved for CBS in this queue. This obviously requires:

TASBand > idleSlope

and:

gateOpenTime > tAllStreams* portTransmitRate / idleSlope

IEEE Std 802.1BA-2011 page 15:



$$\text{Max Latency} = t_{\text{Device}} + t_{\text{MaxPacketSize+IPG}} + (t_{\text{AllStreams}} - t_{\text{StreamPacket+IPG}}) \times \text{Rate/MaxAllocBand} + t_{\text{StreamPacket}}$$

where

$\mathbf{t_{Device}}$ = the internal delay of the device (in increments of 512 bit times)

NOTE 3—t_{Device} is an integral multiple of 512 bit times so that it scales with the speed of the media.

$\mathbf{t_{MaxPacketSize+IPG}}$ = the transmission time for a maximum size interfering frame (1522 octets to 2000 octets) plus its preamble and start of frame delimiter (SFD) (8 octets), and the following inter-packet gap (IPG) (12 octets)

$\mathbf{t_{StreamPacket}}$ = the transmission time for the maximum frame size of the stream that is being reserved, plus its preamble and SFD (8 octets)

$\mathbf{t_{StreamPacket+IPG}}$ = the transmission time for the maximum frame size of the stream that is being reserved, plus its preamble and SFD (8 octets) and the following IPG (12 octets)

Rate = the transmission rate of the port

MaxAllocBand = maximum allocatable bandwidth, the maximum amount of bandwidth the AVB system is able to allocate for Class A streams on the port

$\mathbf{t_{Interval}}$ = the Class A observation interval or 125 µs

$\mathbf{t_{AllStreams}}$ = (MaxAllocBand × t_{Interval}) / tRate = the sum of the transmission times of all Class A stream frames the AVB System is able to allocate in an observation interval (125 µs) on a port

Can be used to derive a lower limit for gateOpenTime:

gateOpenTime ≥ tDevice + tMaxPacketIPG + tStreamPacket +
(tAllStreams – tStreamPacketIPG) * portTransmitRate / idleSlope.......................................(5.2-1)

Plugging in an expression for:

pTR / idleSlope = (pTR * GateOpenTime) / (operIdleSlope * OperCycleTime)

from (1.17-1), yields:

gateOpenTime ≥ tDevice + tMaxPacketIPG+ tStreamPacket +
(tAllStreams – tStreamPacketIPG) * (pTR * gateOpenTime) / (operIdleSlope * operCycleTime)

gateOpenTime * (1 - (tAllStreams – tStreamPacketIPG) * pTR / (operIdleSlope * operCycleTime))
≥ tDevice + tMaxPacketIPG + tStreamPacket

gateOpenTime ≥ (tDevice + tMaxPacketIPG + tStreamPacket) /
(1 – ((tAllStreams – tStreamPacketIPG) * pTR / (operIdleSlope * operCycleTime))) ....(5.2-2)

Assuming operCycleTime is determined by applications not using the CBS and portTransmitRate is a given (taking (1.17-3) into account), this allows to determine a lower limit for gateOpenTime for a given allocated CBS bandwidth (operCycleTime) and usage (tAllStreams) to allow for overprovisioning.

It actually seems appropriate to use overprovisioning from section 1.19, as dynamically adjusting operCycleTime and/or tAllStreams will likely require to change not only gateOpenTime, but also operCycleTime, which may interfere with other applications.

5.2.1 Contiguous Control List Entries

Equation (5.2-2) only holds, if the CBS queue can transmit its frames at the intervals defined and especially directly after an interfering frame from another open TAS gated queue open at the same time. As the TAS gate may decline transmission of a frame if it can not be completed within the current ‘open’ interval (see section 3.3), for any queue controlled by CBS and TAS, the TAS gate control entries shall be configured in such a way that the gate for the CBS controlled queue only opens and closes at most once per TAS cycle. This means the gateOpenTime should not be fragmented across the operCycleTime, but shall be one contiguous time block, even if it extends over multiple control list entries. Otherwise an interfering frame would have to be considered for every ‘open’ gate control entry and the total gateOpenTime would have to increased further.

5.3 Maximum CBS bandwidth with TAS gates enabled

IEEE Std. 802.1Q suggests to only allocate a maximum of 75% of a port’s bandwidth for CBS (see 5.3.1 for a rationale), meaning operIdleSlope ≤ (75% * R), which is equivalent to (tAllStreams / observationInterval) ≤ 75% when operating without the TAS gates.

With the TAS gates enabled, one still can start the configuration (without 1.19) at:

operIdleSlope = cbsBits / observationInterval ....(5.3-1)

From (1.17-3):

operIdleSlope ≤ portTransmitRate * (gateOpenTime / operCycleTime)

One can postulate:

(gateOpenTime / operCycleTime) ≥ operIdleSlope / (75% * portTransmitRate)......................(5.3-2)

This condition can be achieved in two ways:

1) Either (gateOpenTime / operCycleTime) is set to fulfil the desired condition through initial configuration,
2) or the (gateOpenTime / operCycleTime) must be adjusted for every new (de-)reservation of a CBS stream, e.g. based on the information delivered by SRP.

As was already concluded in section 5.2, this document strongly suggests for the implementer to configure (gateOpenTime / operCycleTime) to the maximum required bandwidth for CBS in order to prevent complex re-configuration at runtime.

Using (1.11-1) one can also express (5.3-2) as:

operIdleSlope ≤ 75% * TASBand ....(5.3-3)

Further using the definition from section 1.12 this results in:

MaxAllocBand = 75% * TASBand * (operCycleTime / gateOpenTime)

MaxAllocBand = 75% * portTransmitRate ....(5.3-4)

Which is identical to the original suggestion in IEEE Std. 802.1Q.

5.3.1 One rational for 75% CBS limit

There are multiple more or less arbitrary ways to argue for the limit stated in IEEE Std. 802.1Q section 34.3.1: “The recommended default value of *deltaBandwidth(N)* for the highest numbered traffic class supported is 75%”.

From the automotive perspective one can assume 100Mbit/s links at the time the above statement was drafted. On such a link a 1500Byte max. Ethernet frame takes about 120µsec to transmit (ignoring e.g. IPG for now). The observation interval for SR Class A is 125µsec (driven from the audio use-case), which was at the time presumed to be pervasive.

Doing a very rough approximation one can assume: tMaxFrame ≈ observationIntervalA (at 100Mbit/s), leading to: observationIntervalA * (1 – 75%) * 4 ≈ tMaxFrame
Meaning an interfering max. Frame can be “made up” within a burst of less than 4 observation Intervals (tMaxFrame < observationIntervalA), equivalent to 4 max. Frame sizes and translating into a 4 * 1500byte = 6kbyte maximum buffer requirement for a transmit or receive queue. This was rather arbitrarily considered an acceptable limit.

In further discussions an SR Class C of 1333usec (AVnu AVB Spec) was found to be more efficient. Here 75% * 1333µsec means that observationIntervalC * (1 – 75%) ≈ 3 * tMaxFrame (still on 100Mbit/s links) allows for “make up” of an interfering Frame within the observation interval as well as the use of more than one max. Frame in the CBS shaped streams, which is important for video transport.

5.3.2 An alternate CBS bandwidth limit

Instead of (5.3-3), one could have postulated:

idleSlope ≤ 75% * TASBand ....(5.3.2-1)

operIdleSlope * (operCycleTime / gateOpenTime) ≤ 75% * pTR * (gateOpenTime / operCycleTime)

operIdleSlope ≤ 75% * pTR * (gateOpenTime / operCycleTime)^2

(gateOpenTime / operCycleTime)^2 ≥ operIdleSlope / (75% * pTR) ....(5.3.2-2)

This is obviously a much stronger restriction than the result in (5.3-2), as (gateOpenTime / operCycleTime) < 1, (gateOpenTime / operCycleTime)^2 << (gateOpenTime / operCycleTime).

Without further arguments there seems no good reason to restrict the CBS bandwidth further than what (5.3-2) requires.

5.4 Worst Case CBS Latency with TAS gates enabled

The maximum latency given in IEEE Std 802.1BA-2011 page 15 is:

**maxLatencyBA = tDevice + tMaxPacketIPG+ tStreamPacket +
(tAllStreams – tStreamPacketIPG) * portTransmitRate/MaxAllocBand ....(5.4-1)**

With TAS enabled this should be modified by the gateOpenDelay which is determined by the time it takes for the TAS gate to open:

gateOpenDelay = operCycleTime – gateOpenTime ....(5.4-2)

if contiguous Control List Entries are observed (section 5.2.1). It should be added to (5.4-1), if synchronous frame arrival can not be guarantied (i.e. no TAS Bus Mode nor TAS Phased Mode).

If TAS gates are synchronized across the network as described in sections 3.1 or 3.2 it is important to ensure sufficient timing accuracy and/or guard bands in order to prevent this effect.

The worst case latency can therefore be expressed as:

wcLatencyCBSTAS =
tDevice + tMaxPacketIPG + tStreamPacket + (operCycleTime – gateOpenTime) +
((75% * pTR * gateOpenTime) – streamPacketIPGBits) / (75% * pTR) ....(5.4-3)

with streamPacketIPGBits = tStreamPacket * portTransmitRate and assuming a maximum allocation and usage according to section 5.3.

5.4.1 CBS Latency with TAS gates enabled and using Overprovisioning

Using the definitions in section 1.19 along with (5.3-4), one concludes:

opLatencyCBSTAS =
tDevice + tMaxPacketIPG + tStreamPacket + (operCycleTime – gateOpenTime) +
(cbsBits – streamPacketIPGBits) / (75% * portTransmitRate) ....(5.4-4)

where cbsBits < (75% * pTR * gateOpenTime) as per section 1.19.

5.5 Conclusions on CBS and TAS gate combination

Section 5.1 shows that without interference the configuration of the TAS is forced by the CBS configuration, potentially making integration of other TAS related use-cases difficult.

If interference is allowed, equation (5.3-2) made clear, that a dynamic configuration, e.g. by using SRP, of the combined CBS with TAS gates system is not desirable, as CBS reservation changes would also result in TAS gate schedule changes.

Equation (5.4-3) shows a clear need to have synchronized TAS gate opening times, in order to prevent (operCycleTime – gateOpenTime) to be added to the latency.

It is also important to point out, that in may cases:

tMaxPacketIPG < (cbsBits – streamPacketIPGBits) / (75% * portTransmitRate) ....(5.5-1)

particularly for video transport, where an SR Class with observationInterval > 1ms will likely be used and where cbsBits is potentially an integer multiple of (tMaxPacketIPG * portTransmitRate).

Assuming an arbitrary allocation (idleSlope) and usage (cbsBits) from section 1.19:

maxLatencyCBSTAS =
tDevice + tMaxPacketIPG + tStreamPacket + (OperCycleTime – GateOpenTime) +
(cbsBits – streamPacketIPGBits) / (idleSlope) ....(5.5-2)

shows, that designing a combined CBS/TAS queue to meet a certain latency will be difficult, as it results in a somewhat circular relation, since gateOpenTime determines idleSlope, which in turn influences maxLatencyCBSTAS, which puts a lower limit on gateOpenTime (even ignoring the gateOpenDelay).

Equations (5.2-2) and (1.17-3) can be used to find a static configuration, but are hardly suitable to allow for dynamic changes during run-time.

5.5.1 Alternate approach to CBS with TAS gates

A possible solution could be to choose a configuration where:

operCycleTime ≈ gateOpenTimeCBS .... (5.5.1-1)

Meaning the gate for the CBS queue is closed only for a small time during the operCycleTime in order to allow for another (third) TAS gate to be open for:

gateOpenTimeTAS = (operCycleTime – gateOpenTimeCBS) << gateOpenTimeCBS .... (5.5.1-2)

while the CBS queue along with the interfering queue have their TAS gates closed.

This would lead to:

idleSlope ≈ operIdleSlope .... (5.5.1-3)

Assuming gateOpenTimeTAS ≈ tMaxPacketIPG would allow for a single high priority frame to pass every operCycleTime, taking (5.5-1) into account and further simplifying tStreamPacket ≈ tMaxPacketIPG and cbsBits ≈ (n * tMaxPacketIPG / portTransmitRate), yields:

maxLatencyALT =
tDevice + 3 * tMaxPacketIPG + (n – 1)* tMaxPacketIPG / (pTR * idleSlope) .... (5.5.1-4)

The gateOpenDelayCBS = gateOpenTimeTAS, which is the time the gate is open for in this case a single third TAS stream, has now been reduced to just another interfering frame latency for the CBS stream.

For the TAS stream gateOpenTimeTAS << operCycleTime, meaning it can suffer a significant delay gateOpenDelayTAS = (operCycleTime – gateOpenTimeTAS) if it arrives out of sync. Since gateOpenTimeTAS is now in the order of a max. Frame (tMaxPacketIPG), this may lead to even more stringent requirements on the time-synchronisation, especially on high line rate links.