

ATS token timing

Mick Seaman

ATS (Asynchronous Traffic Shaping) schedulers use a token bucket based algorithm to assign an eligibility time (the earliest permitted transmission time) to each forwarded frame. The time dependent scheduler state is represented by a time, the BucketEmptyTime, rather than a count of tokens held in a bucket. This time representation proves convenient in a number of ways. It supports the assignment of future eligibility times, at which the number of notionally accumulated tokens will be sufficient to forward a given frame, as opposed to using the token bucket algorithm (as commonly described) to simply discard any frame whose transmission would consume more tokens than are presently available. Each frame is the subject of a single invocation of the eligibility assignment procedure, rather than of notionally continuous comparison of its token requirements with those presently in the bucket. All calculations reference times and the time intervals between significant events as measured by a single clock instance: the granularity questions that inevitably accompany a specification based on the notionally continuous addition of tokens to a bucket count are avoided, and replaced with simpler statements about clock accuracy. Delays to frame eligibility due to membership of a scheduler group are easily accommodated.

The IEEE Std 802.1Qcr-2020 (and IEEE P802.1Q-Rev) specification of eligibility time assignment includes, but is not dependent upon, references to the token bucket algorithm and to tokens. The common specification risk in referring to an algorithm that implementors already understand is that they will implement that algorithm (or at least their understanding of it) instead of the specification.

The purpose of this note is to show that IEEE Std 802.1Qcr-2020 is correct in stating that its references to the token bucket algorithm and tokens are purely informative, that the specification would be complete without those references, and that they are not accurate. Their removal should improve the probability of correct implementation.

To demonstrate the independence of the Qcr specification from other descriptions of the token bucket algorithm, this note describes that specification in terms of tokens (with the thorny exception of scheduler group effects). I have not had to refer to any other material to do that, and I do not believe my description is innovative (nor is understanding it necessary for ATS implementation). However it may be useful to anyone who is familiar and comfortable with token bucket algorithm implementations and has not had the time to work out the relationship between BucketEmptyTime and a bucket (a fluctuating count) of tokens. Use the standard, and not this note, when implementing.

1. ATS Scheduler state

The performance of a given ATS scheduler is characterized by its *CommittedInformationRate* and *CommittedBurstSize* parameters.¹ Its state at any given time (t , say), given by the clock that it uses, is a forecast of the difference, Δt , between t and the schedulerEligibility time it will compute on processing a frame of any given length and a forecast of its state once that processing is complete.²

IEEE Std 802.1Qcr [1] uses a *BucketEmptyTime* variable, a reference to a particular time by the scheduler’s clock, as a record of the current state. The *CommittedInformationRate* is customarily specified in bits per second, so it is convenient to express the *CommittedBurstSize* in bits, and the *BucketEmptyTime*, clock times in general, and any interval between two clock times in seconds.

The scheduler state could also be represented by a time dependent number of tokens (denominated in bits for convenience).³ This note describes the relationship between tokens and *BucketEmptyTime* (Figure 1-1), and how both change as the scheduler processes frames (Figure 1-2 through Figure 1-8, which include the temporary variables specified in [1]). In the latter figures a_I is the arrivalTime of a frame I, s_I its schedulerEligibilityTime, and e_I its eligibilityTime. The *lengthRecoveryDuration* and *bucketFullTime* are abbreviated to *lenRd* and *bFT* when convenient.

Figure 1-1 illustrates the relationship between these two ways of expressing the scheduler’s state.

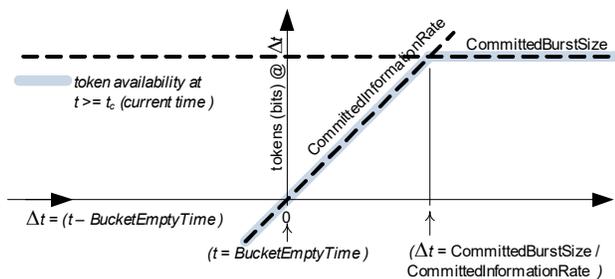


Figure 1-1—BucketEmptyTime and tokens

The scheduler state is only the current picture of token availability, even when scheduler eligibility at other

times is considered. It, and Figure 1-1, is not a historical record. In Figure 1-2, for example, a frame that does not exhaust the immediately available supply of tokens has arrived and been scheduled as immediately eligible for transmission at $a_I = t_I = e_I$, and *BucketEmptyTime* updated appropriately.

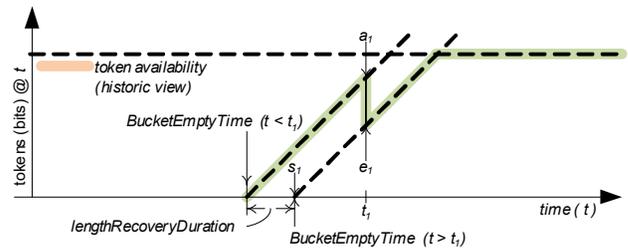


Figure 1-2—An historical account

The notional token bucket **was not empty** at the updated *BucketEmptyTime*,⁴ and an implementation which attempts to record that time by detecting a zero condition for a token count will not perform correctly.

When a frame is scheduled and *BucketEmptyTime* updated, that updated time will be in the past if the frame is eligible for immediate transmission, and in the future if the transmission has to be delayed to allow more tokens to be accumulated.^{5,6}

Negative token counts accommodate scheduling at a future time when the count has recovered sufficiently to accommodate a frame’s length as in Figure 1-3.

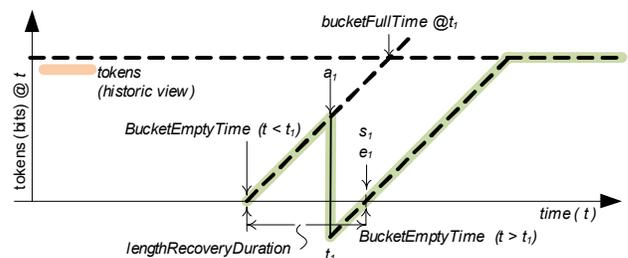


Figure 1-3—Token debt

A number of frames can arrive, and can be eligible for transmission before the token count becomes positive again, just as long as sufficient tokens have been

¹ This note assumes that these are fixed parameters, and does not consider what could or ought to be done when and if they are changed by management.
² These forecasts can also be influenced by prior eligibility time computations by other schedulers in the same Scheduler Group.
³ The use of tokens introduces questions of token size granularity and addition timing, and the accuracy of those parameters. As specified in [1] each invocation of the ProcessFrame(frame) procedure needs only one conversion from bits to time—in the calculation of $lengthRecoveryDuration = length(frame)/CommittedInformationRate$. The calculation of $emptyToFullDuration = CommittedBurstSize/CommittedInformationRate$ used in the determination of *bucketFullTime* need only be performed when one of the latter variables have been changed.
⁴ In other words the description of *BucketEmptyTime* in 8.6.11.3.3 of IEEE Std 802.1Qcr (P802.1Q-Rev/D1.0 pg241 line 24) is untrue. The *BucketEmptyTime* is not “the most recent instant of time at which the token bucket of the ATS scheduler instance was empty, in seconds”. The token bucket was not necessarily empty at that time (if it is in the past), and that time may be in the future.
⁵ With a check that the frame will be eligible for transmission within *MaxResidenceTime*.
⁶ Not quite the full story. Once *GroupEligibilityTime* is taken into account, future eligibility does not necessarily mean a future *BucketEmptyTime*.

added to the bucket between the successively updated *BucketEmptyTimes*. See [Figure 1-4](#).

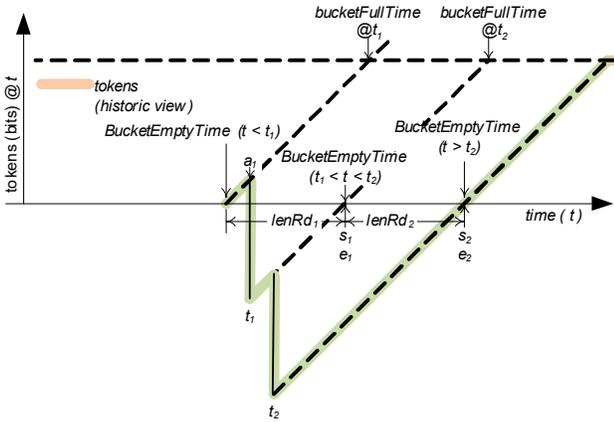


Figure 1-4—Spending while in debt

If a stream has been idle for a while, not using its *CommittedInformationRate*, the scheduler might process a frame and calculate a *schedulerEligibilityTime* that is in the past, before the frame arrived. If a stream was allow to spend unused past tokens it could exceed the committed burst rate, so the *eligibilityTime* is constrained to be no earlier than the frame’s *arrivalTime*, and the *BucketEmptyTime* adjusted ([Figure 1-5](#)) effectively using current tokens.

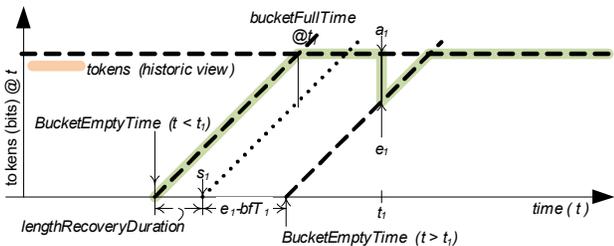


Figure 1-5—Draw from the current account

If the ATS scheduler is part of a scheduler group, a frame’s *eligibilityTime* can be delayed past its *arrivalTime*, even if the scheduler had an ample supply of tokens at that time. The calculated update to *BucketEmptyTime* has the effect of subtracting the necessary tokens at the *arrivalTime*. If the scheduler processes no more frames before the *eligibilityTime* is reached, the resulting number of tokens will be the same as would result from delaying the reception of the processed frame to that time. In [Figure 1-6](#) the calculated *eligibilityTime* is before *bucketFullTime*, sufficient tokens to process the frame are already available, and the number of tokens apparently removed from the bucket corresponds to the length of the processed frame. In [Figure 1-7](#) and [Figure 1-8](#) the updated *eligibilityTime* is after *bucketFullTime*, and the adjustment to *BucketEmptyTime* necessary for the token count at that *eligibilityTime* to reflect the remaining burst capability

appears as a token count reduction at the arrival time in excess of that dictated by the frame’s length.

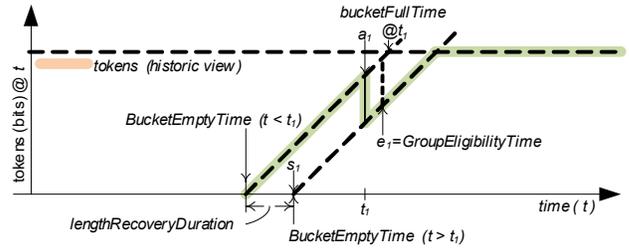


Figure 1-6—Group spending up front

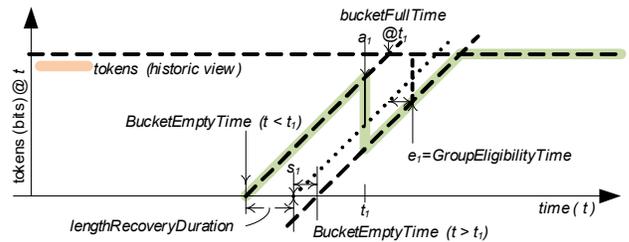


Figure 1-7—Negative interest (1)

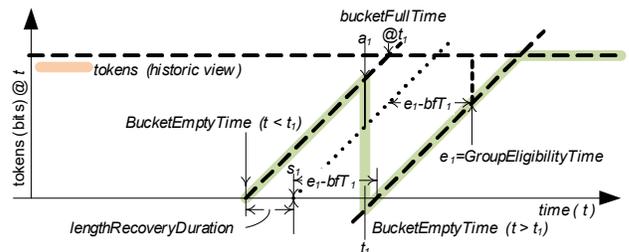


Figure 1-8—Negative interest (2)

2. Recommended P802.1Q-Rev changes

The introduction to this note pointed out that the P802.1Q-Rev/D1.0 ATS scheduler references to the token bucket algorithm are purely informative. I was able to explain the scheduler's calculation of eligibility times in terms of tokens without having to consult any other documents, but the actual calculations specified in P802.1Q-Rev do not use or depend on tokens, and in particular have no notion of token size/granularity. The calculations does not depend on a continuous updating of a token count, and allow a single invocation of a procedure to determine an eligibility time.

The common specification risk in referring to an algorithm that implementors already understand is that they will implement that algorithm (or at least their understanding of it) instead of the specification. In the present case, the use of negative token counts to explain scheduler behavior may not be obvious. Alternative interpretations (leading to the implementation of a different algorithm) might result from the current P802.1Q-Rev references to the token bucket. Some of these references are, at least in my mapping of the specification to a token based explanation, incorrect (see above).

I recommend the removal of the token bucket references, with text changes to P802.1Q-Rev/D1.0 as follows:

- 1) pg 239, line 19: Delete the paragraph referencing Tanenbaum and Wetherall.
- 2) pg 241, lines 24 through 29. Replace the text of 8.6.11.3.3 BucketEmptyTime with the following:

“A variable that embodies the current state of the ATS scheduler. This variable is initialized with a time earlier than CommittedBurstSize/CommittedInformationRate in the past, as perceived by the ATS Scheduler Clock.”
- 3) pg 241, lines 31 and 32. Replace text of 8.6.11.3.4 bucketFullTime with the following:

“A temporary variable used when updating BucketEmptyTime after determining a frame's schedulerEligibilityTime. The prior value of BucketEmptyTime plus the emptyToFullDuration.”
- 4) pg 241, lines 34 through 36. Delete the second sentence of the text of 8.6.11.3.5 CommittedBurstSize.
- 5) pg 242, lines 2 through 4. Delete the second sentence of the text of 8.6.11.3.6 CommittedInformationRate.
- 6) pg 242, line 14 and 15. Replace the text of 8.6.11.3.9 emptyToFullDuration with the following:

“The duration CommittedBurstSize/CommittedInformationRate.”
- 7) pg 242, line 24. Replace the text of 8.6.11.3.12 lengthRecoveryDuration with the following:

“The duration length(frame)/CommittedInformationRate.”
- 8) pg 242, lines 32 and 33. Replace the text of 8.6.11.3.14”

“A temporary variable used in the calculation of a frame's eligibilityTime. The prior value of BucketEmptyTime plus the lengthRecoveryDuration for the frame.”
- 9) pg 2159, line 14. Replace “token-bucket traffic envelopes” with “traffic envelopes”.
- 10)pg 2161, lines 13 and 14. Replace “token-bucket envelope enforcement” with “envelope enforcement”.
- 11)pg 2168. Delete bibliographic reference [B77].

The proposed changes to 8.6.11 are shown in attachments to this note. The “...-changes” file shows the changes using the normal amendment strikethrough and insert conventions. The “...-updated” file shows the end result after the changes have been applied.