P802.1Qcc D1.0 Comment #47

CRAIG GUNTHER

13JUNE2016

THE NOTE IN QUESTION

Clause 35.2.2.8.6 Accumulated Latency, page 51, lines 28-31:

> NOTE—The behavior of reporting of failure upon a change in Accumulated Latency is specific to systems that implemented the original ProtocolVersion 0x00 of MSRP. Implementations of the ProtocolVersion of MSRP in this standard report a failure when the analogous AccumulatedLatency exceeds NetworkRequirements.MaxLatency, as specified in 35.2.4.6.

My initial concern: This change creates behavior that is incompatible with the original MSRPv0. Existing AVB networks set their 1722 presentation time (when the A/V samples are played) based on the Accumulated Latency that is reported by the network. If that latency were to increase after the stream starts "playing" the A/V FIFOs in the Listeners would underrun and the quality of the A/V would degrade. MSRPv0 prevents this from occurring by changing the Talker Advertise to a Talker Failed if the Accumulated Latency changes.

The Editor has agreed (ACCEPT IN PRINCIPLE) with my concern with regard to MSRP-based reservations; but what about reservations created with the UNI?

COMMENT #47

MY BALLOT COMMENT

"The behavior described in this NOTE is not exactly what I would have expected. We need to agree upon the behavior before I can suggest a change.

I have two expectations:

1. *If NetworkRequirements.MaxLatency is assigned a value and AccumulatedLatency exceeds it, a FailureCode 21 will be used.*
2. *If NetworkRequirements.MaxLatency is not provided (see pg 59, ln 19) or NetworkRequirements.MaxLatency is not defined (i.e. ProtocolVersion 0x00) then if AccumulatedLatency increases beyond its initial value (see wording in 35.2.2.8.6), a FailureCode 7 will be used.*

If either of these FailureCodes are encountered then the TalkerAdvertise should be propagated as a TalkerFailed with that code. I don't believe this requires any changes to Table 99-1 (Failure Codes)."

COMMENT #47

EDITOR'S RESPONSE

"ACCEPT IN PRINCIPLE.

For the sake of discussion, let's split the "or" in item 2 into items 2 and 3, since they are distinct.

Qcc D1.0, covers item 1 and 3, but item 2 is inconsistent.

According to the specs in clause 99, NetworkRequirements.MaxLatency is a requirement, meaning that if it is not provided, it does not apply. StreamStatus.AccumulatedLatency is status, meaning the currently configured worst-case latency.

MSRPv0 combined these concepts, in that the very first AccumulatedLatency transmitted in an MSRPDU is stored as an implicit "MaxLatency", and if subsequent AccumulatedLatency ever exceeds that value, the stream fails.

Qcc D1.0 maintains MSRPv0 behavior for MSRPv0, but the commenter wants to retain this behavior for MSRPv1 if/when the MaxLatency value is not provided (i.e. no requirement). The rationale is that an AVB product may replace an MSRPv0 stack with an MSRPv1 stack without providing the explicit MaxLatency, in which case the "implicit" MaxLatency must be applied for compatibility.

The editor proposes to make this change for SRP (clause 35), but leave the behavior as-is in clause 99 (separate concepts). The SRP behavior can be specified as a failure check that is specific to SRP, layered on top of clause 99 specs."

COMMENT #47

My interpretation of Editor's response

Split my comment into three parts:

1. MSRPv1: If AccumulatedLatency > MaxLatency; then FailureCode=21.
2. MSRPv1: If MaxLatency is not provided and AccumulatedLatency increases;
   a. If SRP (clause 35); then FailureCode=7 (MSRPv0 compatible behavior),
   b. If UNI (clause 99); then no FailureCode.
3. MSRPv0: If AccumulatedLatency increases; then FailureCode=7.

The only concern I have is with 2b.

My interpretation of Editor's Response: *"If there is no MaxLatency specified then there is no limit placed on the AccumulatedLatency and therefore no TalkerFailed*". I would concede that the Editor makes a valid argument; however, please consider…

COMMENT #47

UPGRADE STRATEGY

Qcc is adding a lot of capabilities that will be useful for users of SRP. Introducing TLVs in MSRPv1 provides a backwards compatible upgrade path to MSRPv0 users (see Qcc D1.0 35.2.4.3.1).

As a provider of SRP-based solutions I will encourage our customers to introduce MSRPv1 capable bridges into the network to get TSN features such as Qch shaping, CB redundancy, Qcc MaxLatency and Talker Pruning capabilities, AS redundancy, etc. I will also encourage end-station providers to migrate to MSRPv1 to enable these new TSN capabilities.

I would anticipate that the upgrade strategy would involve a staged rollout such that critical areas of the network core are the first devices to add support for TSN features. This could produce "segments" of TSN bridges connecting "segments" of AVB bridges and end-stations. End-stations needing TSN features would be upgraded next. Finally the rest of the network core and other end-stations would be upgrade, although this last step would likely be done very slowly because of financial considerations ("*if it ain't broke, don't fix it*").

Now let's talk about my concern…

COMMENT #47

UPGRADE PATH CONCERNS

Upgrading SRP (non-UNI) networks from MSRPv0 to MSRPv1 involves a logical progression that developers and customers can easily understand. QoS will remain at the current high levels that we (IEEE 802.1) have embraced since AVB began. I can easily "sell" this concept to anyone who is interested.

My concern is that during this upgrade process we will likely need to introduce segments of the network that must be manually configured with a UNI for features such as CB (redundancy). The UNI may also be used to configure the core of large networks, allowing SRP to run on edge segments.

So what will happen to the QoS of these networks when some portions of the network are running SRP and treat MaxLatency one way and other portions of the network are configured by the UNI and treat MaxLatency a different way (slide 5 item 2a vs. 2b)? Likewise what happens if an end-station switches to UNI control when it was running SRP previously?

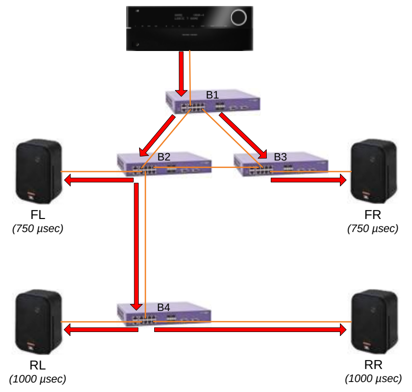
I don't not want to allow a situation to occur where we are no longer certain about the QoS since the default behavior of MSRPv1 with a UNI will be no latency guarantees unless EVERY end-station declares a MaxLatency.

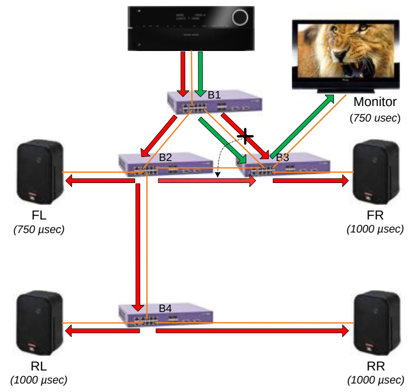
How did we get into this situation? …

COMMENT #47

WHERE I THINK THE PROBLEM STARTED

March 2014 (Beijing) I presented* the concept of AcceptableLatency:





In this example the FL & FR speakers inform the network they can accept latency up to 1000 usec and still play in sync with the RL & RR speakers. This allows the network to reconfigure the audio path to the FR speaker when the video reservation is established. The RL & RR speakers still require the latency guarantees provided by MSRPv0.

In that same presentation I also explained uses where a single A/V path can be made up of multiple AVB streams between processing devices, or of a stream going to a recording truck that could accept much larger latency than the A/V devices in the concert hall.

The intent of AcceptableLatency was to allow a subset of end-stations to give the network flexibility in reconfiguring stream paths.

Unfortunately AcceptableLatency was turned into MaxLatency when we introduced the concept of MinLatency. MinLatency has since disappeared and MaxLatency has taken on a new meaning (to me).

* See: cc-cgunther-acceptable-latency-0314-v01.pdf

COMMENT #47

PROPOSED SOLUTION

Behavior of reservations should be identical regardless of how they are configured. It shouldn't matter if the path(s) between a Talker and its Listener(s) are configured via SRP, or the UNI, or a combination of SRP and the UNI.

Behavior of reservations should be backwards compatible with AVB's MSRPv0.

Therefore AccumulatedLatency should have the same affect on TalkerAdvertise transitioning to TalkerFailed in MSRPv1 as it did in MSRPv0 when no *AcceptableLatency* (i.e. NetworkRequirements.MaxLatency) is declared. If an MSRPv1 device declares an *AcceptableLatency* then that new behavior should be enforced.

So how would a new MSRPv1 device declare that it doesn't care about latency and AccumulatedLatency should be allowed to increase without setting a TalkerFailed? My recommendation is to explicitly declare MaxLatency with as large a value as possible. This would apply to MSRPv1 whether using SRP or the UNI.

Here's the rules I would propose (which guaranteed backward compatibility):

- MaxLatency not declared implies AccumulatedLatency works as it does in MSRPv0.
- MaxLatency set to a maximum value implies any AccumulatedLatency is acceptable.
- MaxLatency set anywhere in between implies a corresponding limit to AccumulatedLatency.

Regardless of how the MaxLatency attribute got there (SRP vs. UNI), it should be treated consistently internally.

COMMENT #47

Thanks!