MaxLatency Contribution

Astrit Ademaj

Introduction

- The issue of transforming the semantics of *MaxLatency* parameter from “Latency-semantics” to “Deadline-semantics” for time-aware streams was discussed in the previous calls.
  - This contribution proposes adding of an additional parameter to address the “deadline requirements”

- In addition, there is an issue with the *MaxLatency* when this parameter is used with the „Latency-semantics” and specially in combination with preemption
  - This presentation describes the issues with the existing MaxLatency definition and proposes 3 different options (contributions) to address the given issue

- This contribution does not address the issues with the definition of the “*reference points*”.

IEEE 802.1Q and 802.1Qcc Definitions

3.118 Latency – is currently defined as:

> **The delay experienced by a frame in the course of its propagation between two points** in a network, measured from the time that a known reference point in the frame passes the **first point** to the time that the reference point in the frame passes the **second point.**

IEEE 802.1Qcc specifies a MaxLatency model and the CNC-CUC interface, and defines different reference points for:

- non-time-aware streams
  - 46.2.3.6.2 Latency shall use the definition of 3.118, with additional context as follows: The 'known reference point in the frame' is the message timestamp point specified in IEEE Std 802.1AS for various media (i.e. start of the frame). The '**first point**' is in the Talker, at the reference plane marking the boundary between the network media and PHY (see IEEE Std 802.1AS). The '**second point**' is in the Listener, at the reference plane marking the boundary between the network media and PHY.
- time-aware streams and
  - 46.2.3.6.2 When TSpecTimeAware is present: The '**first point**' is assumed to occur at the start of the Interval, as if the Talker's offsets (EarliestTransmitOffset and LatestTransmitOffset of 46.2.3.5) are both zero.

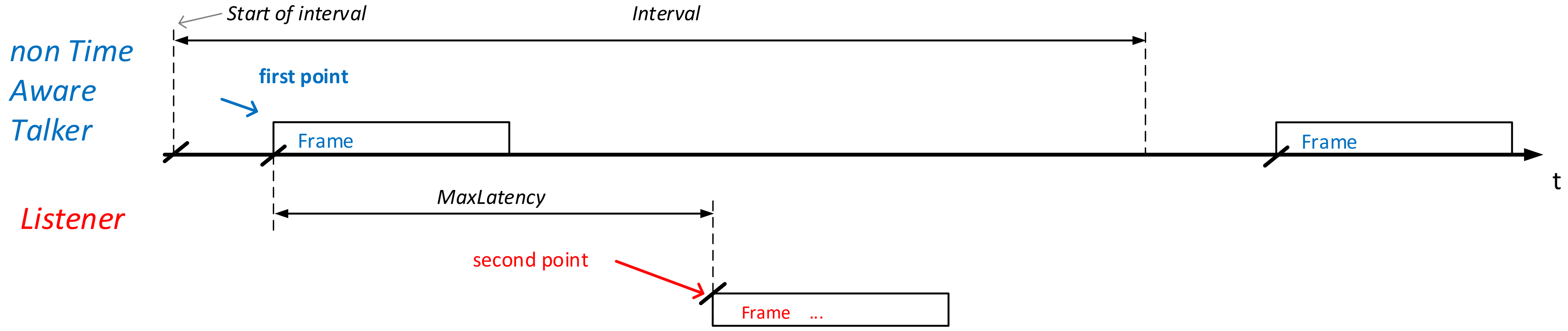
MaxLatency is a request parameter to the CNC for a particular stream (requirement to CNC for a particular stream set up by a listener*)

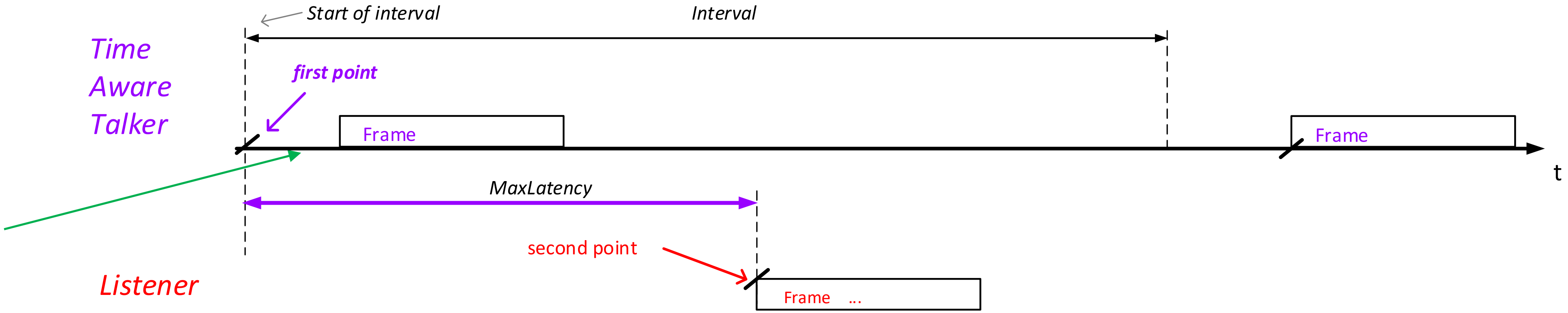
Issue: “MaxLatency” definition for time aware talkers (1)

§3.118 Latency: The delay experienced by a **frame in the course of its propagation** between two points in a network, measured from the time that a known reference point in the frame passes the first point to the time that the reference point in the frame passes the second point.

46.2.3.6.2 When TSpecTimeAware is present: The ’first point’ is assumed to occur at the start of the Interval, as if the Talker’s offsets (EarliestTransmitOffset and LatestTransmitOffset of 46.2.3.5) are both zero.

MaxLatency definition becomes a “**deadline**” semantics for a time-aware talker





There are time intervals where the frame is not in propagation at all

Issue: “MaxLatency” definition for time aware talkers (2)

- *MaxLatency definition* is “transformed” to a “*Deadline” definition* for “time-aware” streams
  - *Latency requirements/constraints* and “*Deadline” requirements/constraints* are different values
  - With the existing definition there is possibility to express requirements with latency semantics for time-aware streams

Proposal 1: Introduce an explicit parameter for :

- **“deadline” requirement for listeners and**
- **“deadline” response for listeners**

Contribution: “Deadline” definition for time aware talkers

§46.2.3.6.xy *MaxDeadline* – is defined as:

The point in time within ‘Interval’ where the end of the last symbol of the FCS passes the reference plane (PHY).

MaxDeadline is a request parameter in the CNC-CUC interface

§46.2.3.6.xy *ListenerDeadline* – is defined as:

The point in time within ‘Interval’ where the end of the last symbol of the FCS passes the reference plane marking the boundary between the network media and PHY.

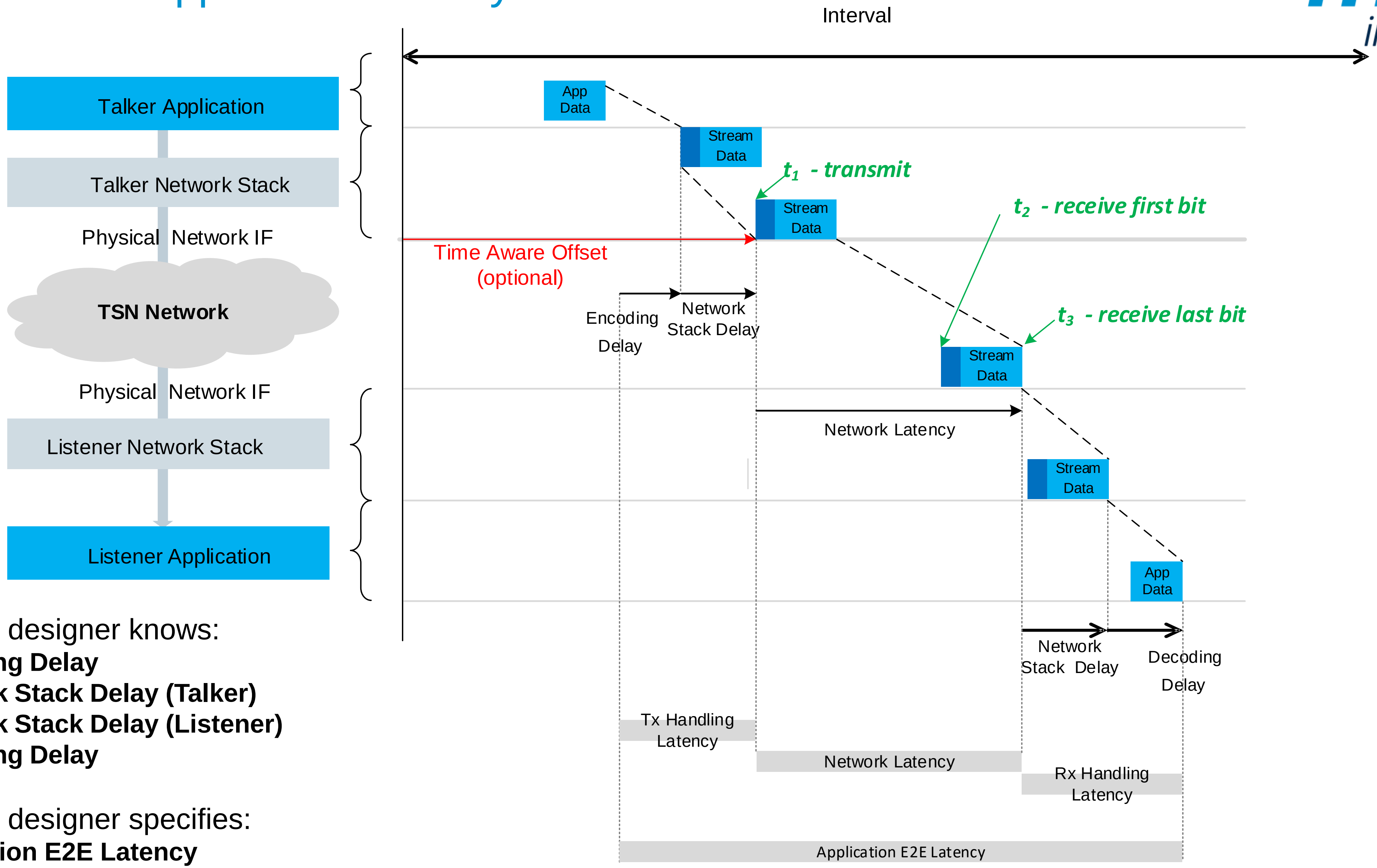
ListenerDeadline is the response parameter in the CNC-CUC interface. ListenerDeadline shall be equal or smaller than MaxDeadline

In case that *MaxDeadline* or *ListenerDeadline* value is larger than the *Interval* value, that means that the frame is being issued by the talker in the previous interval, and the frame is being expected in the following interval at the:

Phase offset = *ListenerDeadline* **%** *Interval*

MaxLatency issue

End-To-End Application Latency



Application designer knows:

- **Encoding Delay**
- **Network Stack Delay (Talker)**
- **Network Stack Delay (Listener)**
- **Decoding Delay**

Application designer specifies:

- **Application E2E Latency**

Application Model

- Distributed control application
- TSN Network is designed to use a CNC (*fully centralized model* is assumed*)
- Application designer knows (endsystem impementation specific parameters):
  - **Encoding Delay**
  - **Network Stack Delay (Talker Side)**
  - **Network Stack Delay (Listener Side)**
  - **Decoding Delay**

- Application designer specifies **Application E2E Latency**
- Application designer calculates **Network Latency** from the **Application E2E Latency**
- Application designer does not need to know which (TSN) network features are used

Application E2E Latency is the time interval between:

- the timepoint when the talker application finishes the computation/generation of the stream data and
- the point in time when the stream data are available for the listener application layer

Network Latency is the time interval between

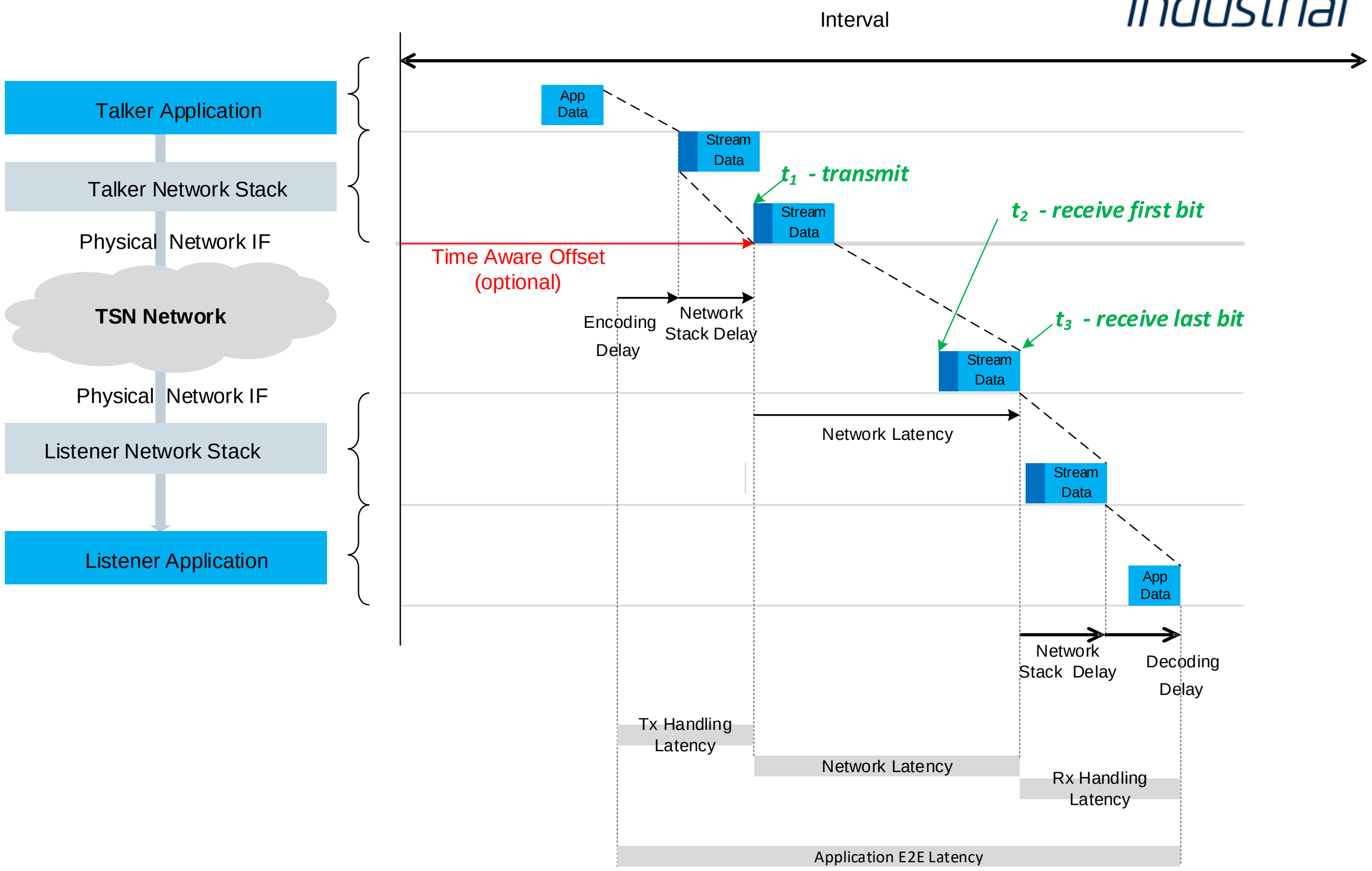
- the timepoint when the first bit „hits“ the reference point (PHY boundary) at the talker
- the timepoint when the last bit leaves the reference point (PHY boundary) at the listener

Configuration Workflow

1. Application designer specifies stream parameters (e.g., talker, list of listeners MaxFrameSize, Interval,…)
2. Application designer specifies the **Application E2E Latency** for listeners
3. Application designer calculates the **NetworkLatency** by using **Application E2E Latency** as basis

```
NetworkLatency = ApplicationE2ELatency
      - EncodingDelay - NetworkStackDelay(Talker)
      - NetworkStackDelay(Listener) + DecodingDelay
```

4. Application designer provide among other parameters (e.g., Max Frame Size) the **NetworkLatency** for the „function“ that sends the request to the CUC/CNC



NetworkLatency = t_3 - t_1 *is the parameter of interest for the application designer*

MaxLatency = t_2 - t_1 *is the available parameter in 802.1Qcc (i.e., it has a different semantics than the NetworkLatency)*

Configuration Workflow (2)

1. Application designer specifies stream parameters (e.g., talker, list of listeners MaxFrameSize)
2. Application designer specifies the ***ApplicationE2ELatency*** for listeners
3. Application designer calculates the ***NetworkLatency*** by using ***ApplicationE2ELatency*** as basis
4. Application designer provide among other parameters (e.g., Max Frame Size) the **NetworkLatency** for the „function“ that sends the request to the CUC/CNC

5. **NetworkLatency** needs to be translated in a parameter available in the 802.1Qcc. i.e., *MaxLatency*
   a) Translation from the **NetworkLatency** to the **MaxLatency** can be done if the listener link speed is known to the application
   b) *In case the preemption is active - this translation shall be done as if the frame is received in one piece.*

MaxLatency (Listener) = func(NetworkLatency,
Listener_link_speed,
MaxFrameSize,
NO_PREEMPTION,
SINGLE_FRAME_PER_INTERVAL)

Possible solutions (contributions)

Option 1: Keep the *MaxLatency* semantics as it is and add additional notes to clarify how the *MaxLatency* and the *AccumulatedLatency* shall be used (interpreted at the CNC/CUC and the Listener)

Option 2: Redefine the semantics of the *MaxLatency*

Option 3: Define additional Latency parameter for time-aware streams

Contribution Option 1: **Keep the MaxLatency semantic (1)**

In this case the application designer, the CNC and the Listener application shall do following calculations

1. Application designer defines *NetworkLatency* for each Talker-Listener pair
   a) translates the *NetworkLatency* to *MaxLatency* for each Talker-Listener pair

2. CNC receives *MaxLatency* as request parameter for stream configuration
   a) *CNC calculates the NetworkLatency based on the MaxLatency, ListenerLinkSpeed, and MaxFrameSize (known at the CNC) as in case that no preemption is in place*
   b) *NetworkLatency* is the parameter of interest for the CNC
   c) In case of multiple frames per Interval, CNC includes in the *NetworkLatency* the talker-jitter as well as IFG

Contribution Option 1: **Keep the MaxLatency semantic (2)**

3. CNC calculates the *AccumulatedNetworkLatency* (first bit transmitted - to last bit received)
   a) *AccumulatedNetworkLatency* shall also cover possible delays caused by frame preemption
   b) As *AccumulatedNetworkLatency* parameter is not available in Qcc model, the CNC calculates the *AccumulatedLatency* based on *AccumulatedNetworkLatency* as in case of no preemption is in place (i.e., reduces the value for the time interval for the transmission of the frame of MaxDataSize and the LinkSpeed)
   c) CNC passes the *AccumulatedLatency* to its listeners through the CUC

4. Listener receives the *AccumulatedLatency* from the CUC/CNC
   a) Listener calculates the *AccumulatedNetworkLatency* (this is the parameter of interest for the listener) based on *AccumulatedLatency* under the assumption that
      - no preemption was used and
      - the listener application knows its link speed

Contribution Option 1: **Keep the MaxLatency semantic - Example**

Example: 100 Mbit/s link speed, MaxFrameSize = 1518 bytes, 123µs is the time needed for this frame to pass over a single link

1. Application designer specifies *NetworkLatency* for a given listener = 2000µs
   a) translates the *NetworkLatency* to *MaxLatency* = 2000µs -123µ = 1877µs
2. CNC receives *MaxLatency* as request parameter 1877µs
   a) *CNC derives the NetworkLatency based on the MaxLatency* 1877µs + 123µs = 2000µs
3. CNC generate its schedule and calculates the *AccumulatedNetworkLatency* which includes delays caused by preemption
   a) CNC can calculate that the frame is beeing preempted and that the first bit is received after 1000µs, but the last bit is received after 1800µs (meaning *AccumulatedNetworkLatency* = 1800µs)
   b) CNC *derives* the *AccumulatedLatency* as 1800µs -123µs = 1677µs
   c) CNC passes the *AccumulatedLatency* value of 1677µs to the listener through the CUC
4. Listener receives the *AccumulatedLatency* from the CUC 1677µs
   a) Listener calculates the *AccumulatedNetworkLatency* 1677µs +123 µs = 1800µs

Summary Contribution Option 1

1. Add the text for additional calculations for *MaxLatency* and *AccumlatedLatency in case of preemption* at the CNC and the listeners
2. Change the semantics of *MaxLatency* for time-aware streams
3. Add the definition for *MaxDeadline* and *NetworkDeadline*

Contribution Option 2: **Change the semantics of MaxLatency**

Change the definition of ***MaxLatency*** to cover the parameter ***NetworkLatency*** without the need to perform calculation at the CNC and at the Listener. Change the semantics from:

- *the time interval from the first bit at the talker to the first bit at the listener*
- *the time interval from the first bit at the talker to the last bit at the listener*

Proposal: change Sect. 46.2.3.6.2 to:

"MaxLatency *shall use the definition of 3.118, with additional context as follows: The "known reference point in the frame" is the message timestamp point specified in IEEE Std 802.1AS for various media (i.e., start of the frame) for the "first point",*

and the end of the last symbol of the FCS for the "second point".

The "first point" is in the Talker at the reference plane marking the boundary between the network media and PHY (see IEEE Std 802.1AS). The "second point" is in the Listener at the reference plane marking the boundary between the network media and PHY."

Summary Contribution Option 2

1. Change the semantics of *MaxLatency* and *AccumlatedLatency*
2. Change the semantics of *MaxLatency* for time-aware streams
3. Add the definition for *MaxDeadline* and *NetworkDeadline*

Contribution Option 3: **Define additional Parameters in Qdj**

- MaxLatency can be keept for non-time aware streams
- Define additional parameters with the semantics of first bit *the first bit at the talker and the last bit at the listener*
- *NetworkMaxLatency (Request parameter to the CNC)*
- *NetworkAccumulatedLatency (Response parameter from the CNC)*

Proposal: change Sect. 46.2.3.6.2.xy to:

NetworkMaxLatency and NetworkAccumulatedLatency shall use the definition of 3.118, with additional context as follows: The "known reference point in the frame" is the message timestamp point specified in IEEE Std 802.1AS for various media (i.e., start of the frame) for the "first point",

and the end of the last symbol of the FCS for the "second point".

The "first point" is in the Talker at the reference plane marking the boundary between the network media and PHY (see IEEE Std 802.1AS). The "second point" is in the Listener at the reference plane marking the boundary between the network media and PHY."

Summary Contribution Option 3

1. Add the definition for *NetworkMaxLatency* and *NetworkAccumulatedLatency*
2. Add the definition for *MaxDeadline* and *NetworkDeadline*

Summary

Option 1: • **Add text for extra calculations** • **Change the semantics for MaxLatency and time-aware streams** • **add MaxDeadline and NetworkDeadline**		**Option 2** • **Change the semantics (MaxLatency and AccumlatedLatency)** • **add MaxDeadline and NetworkDeadline**		**Option 3** • **Define additional parameters** • **add MaxDeadline and NetworkDeadline**	
Pros	**Cons**	**Pros**	**Cons**	**Pros**	**Cons**
---	Calculations need to be performed by the listener application	User Friendly: No calculations need to be performed by the listener application	---	Clear interface	---
---	Listener application **need** to know network details such as link speed	Listener application **does not need** to know network details such as link speed	---	Listener application **does not** need to know network details such as link speed	---
---	Can be confusing in case of switched endstations with an internal link between the switch and the endstation part. Listener application needs to know the internal link speed	In case of switched endstations with an internal link, the internal link speed is known by the CNC, transparent to the listener application and application designer	---	In case of switched endstations with an internal link, the internal link speed is known by the CNC, transparent to the listener application and application designer	---
No backward compatibility issues for non-time-aware streams	Possible backward compatibility issues for **time-aware streams**	---	Possible backward compatibility issues for **time-aware** and **non-time aware** streams	No backward compatibility issues	---