LNI 4.0 Contribution to IEEE P802.1Qdd D0.7 Template support based on a Time Aware Shaper Traffic Class

The aim of this document is to highlight the contribution points into IEEE P802.1Qdd D0.7 document from the point of LNI 4.0 community. LNI 4.0 utilizes two RA classes regarding "Time Aware Shaper" (TAS) algorithm aka "Enhancements for Scheduled Traffic" (EST).

1. **Contribution Item: Template defined additional boundary port rules**
   - **Existing Situation in LNI 4.0**
     - The LNI4.0 data plane uses EST therefore the dataplane depends on correct alignment of windows per class. This there are cases where there is the correct priority and even same RA Class type on both sides but there's still no way a stream may pass.
       Note: The window configuration is subject to a later point in this document

   - **Current Proposal in IEEE P802.1Qdd D0.7**
     - IEEE P802.1Qdd D0.7 does decide on boundary port or not only by presence of an RA class with same priority

   - **Contribution in IEEE P802.1Qdd D0.7**
     - Suggestion-1: Class-Template specific extension to allow a class to add additional reasons for a port to be a boundary port could be added

Page 45 51.3.2.4 Line 26

Add:

Depending on the RA Class definition additional reasons for a port to be a boundary port can be specified.

Page 95 51.8.5.14 Line 33

Change:

```
if (tNeighborRaClass != NULL &&
        tNeighborRaClass.priority == tLocalRaClass.priority) {
```

➔

```
if (tNeighborRaClass != NULL &&
        ( tNeighborRaClass.priority == tLocalRaClass.priority &&
        getRAClassSpecificBoundary(tLocalRaClass, tNeighborRaClass) == FALSE) {
```

Add:

51.8.5.22 getRaClassSpecificBoundary(tLocalRaClass, tNeighborRaClass)

This procedure evaluates tLocalRaClass and tNeighborRaClass to provide additional reasons for a boundary. It's behavior is defined per RA Class ID and return TRUE if the evaluation results in a boundary state.
If the RA Class definition does not specify an extended boundary evaluation the return value shall be FALSE.

2. **Contribution Item: Time Aware Scheduler (TAS) RA Class Attributes**

- **Existing Situation in LNI 4.0**
  - LNI 4.0 implement an LNI-SR organizationally specified extension to the RAclass attribute definition via LNI_SR_Enhanced_RAclass_Descriptor sub-TLV.
  - LNI 4.0 achieves the exchange of organizationally specified extension via LNI_SR_Enhanced_RAclass_Descriptor sub-TLV

- **Current Proposal in IEEE P802.1Qdd D0.7**
  - IEEE P802.1Qdd D0.7 introduce **RaClassTemplateDefinedData** to exchange RA class template information.
  - This allows every Bridge in an RA class domain to know the RA class template and the corresponding parameter settings used by its adjacent station(s) for configuration of the RA class.

- **Contribution in IEEE P802.1Qdd D0.7**
  - Instead of using the very specific template defined data add optional tlvs to the RA Class attribute itself. These TLV's could cover the required information for all window and thus time based RA classes
  - Proposed structure for Class **WindowTLV** could be added into **51.5.2.1**
    e.g. 51.5.2.1.5 Window TLV as shown in the following figure.

Name	Range	Meaning	Octet	Length
RaClassCycleTime	1 – 1.000.000.000	Cycle time in units if 1ns	1	
RaClassWindow Offset	0 – 1.000.000.000	Window offset in units of 1 ns		
RaClassWindowLenght	0 – 1.000.000	Window length in units of 1 ns		

  - Proposed structure for Class **ClockTLV** could be added into **51.5.2.1**
    e.g. 51.5.2.1.6 Clock TLV as shown in the following figure.

Name	Range	Meaning	Octet	Length
RaClassClockDomainId		Clock domain ID used for this RA class	1	
RaClassClockPrecisionRequirement	0 – 1.000.000.000	Window offset in units of 1 ns		

3. **Contribution Item: Template defined Latency Adjustment**

- **Existing Situation in LNI 4.0**
  - LNI 4.0 uses specific window based classes with special latency calculation due to it's constraint to linear topology

- **Current Proposal in IEEE P802.1Qdd D0.7**
  - IEEE P802.1Qdd D0.7 does have a fixed definition of how to update latency which does not allow the optimizations possible due to the additional constraint introduced by LNI 4.0

- **Contribution in IEEE P802.1Qdd D0.7**

Page 95 51.8.5.14 Line 52

Change:

```
adjustAccumulatedLatencies(pTaDec) {
        tPort = pTaDec.portRef;
        tRaClass = getLocalRaClass(pTaDec.attr.priority);
        tCurrentHopMaxLatency = portRaClass[tPort, tRaClass.id].maxHopLatency;
        tCurrentHopMinLatency = minProcessingDelay + port[tPort].minPropagationDelay +
        pTaDec.attr.MinTransmittedFrameLength*8*1e9 / port[tPort].portTransmitRate;
        pTaDec.attr.AccumulatedMaximumLatency += tCurrentHopMaxLatency;
        pTaDec.attr.AccumulatedMinimumLatency += tCurrentHopMinLatency;
        }
```

➔

```
adjustAccumulatedLatencies(pTaDec) {
        tRaClassLocal = getLocalRaClass(pTaDec.attr.priority);
        tRaClassNeighbor = getNeighborRaClass(pTaDec.portRef, pTaDec.attr.priority);
        adjustAccumulatedLatenciesRaClass( tRaClassLocal.id, taRaClassLocal, tRaClassNeighbor )
        }
```

Add:

51.8.5.x adjustAccumulatedLatenciesRaClass(RaClassID, RaClassLocal, RaClassNeigbor)

This procedure shall adjust the accumulated latency values based on Ra Class type and information. If a template does not specify a custom calculation the default calculation shall be used.

The lower table specifies the adjustments to be done based on RaClassID

00-80-C2-xx, 00-80-C2-yy	Use Default

Default	tPort = pTaDec.portRef;
	tCurrentHopMaxLatency = portRaClass[tPort, tRaClass.id].maxHopLatency;
	tCurrentHopMinLatency = minProcessingDelay + port[tPort].minPropagationDelay + pTaDec.attr.MinTransmittedFrameLength*8*1e9 / port[tPort].portTransmitRate;
	pTaDec.attr.AccumulatedMaximumLatency += tCurrentHopMaxLatency;
	pTaDec.attr.AccumulatedMinimumLatency += tCurrentHopMinLatency;

4. Contribution Item: Ta Ingress Processing based on neighbor information

- **Existing Situation in LNI 4.0**

    - Depending on the RaClass information received from the neighbor a talker may or may not be forwarded. E.g. if the required clock precision is not identical.

- **Current Proposal in IEEE P802.1Qdd D0.7**

    - IEEE P802.1Qdd D0.7 51.8.5.2 does only consider presence of am RAClass for talker status

- **Contribution in IEEE P802.1Qdd D0.7**

  Page 91 51.8.5.2 Line 19

  Add:

  ```
  pTaReg.ingressFailureCode = initialize to error;
  ```

  Page 91 51.8.5.2 Line 23

  Add:

  ```
  processTAIngressTemplateErrorCode(pTaReg, tLocalRaClass, tNeighborClass) == TRUE
  ```

  Page 91 51.8.5.2 Line 29

  REMOVE

  ```
  pTaReg.ingressFailureCode = << Failure code TBD >>;
  ```

  Page 91 51.8.5.2 Line 32

  Add:

  ```
  pTaReg.ingressFailureCode = NO_ERROR;
  ```

  Add:

51.8.5.x processTAIngressTemplateErrorCode(paTaReg, tLocalRaClass, tNeighborClass)

This procedure shall evaluate contraints resulting from the local and remote neighbor class configuration and return FALSE if this talker may not be forwarded.

It shall set the error code in pTaReg accordingly.

Note: make sure the error code is not over

8. Contribution Item: Checking Reservability

- **Existing Situation in LNI 4.0**
  - The maximum number of acceptable frames per egress port **(IniSrMaxFrames)** must be calculated and checked by LNI-SR to assure that the assumptions for the calculation of IniSrMaxInterference.
  - IniSrMaxFrames is calculated as the integral number of maximum sized frames, which can be transmitted as burst within **IniSrMaxInterference** [ns].

IniSrMaxFrames =

a) at 1 Gb/s links:

____IniSrMaxInterference / (raClassPortMaxTransmittedFrameSize * 8)

b) at 100 Mb/s links:

____(IniSrMaxInterference / (raClassPortMaxTransmittedFrameSize * 8)) / 10

- **Current Proposal in IEEE P802.1Qdd D0.7**
  - **99.8.5.16 checkReservability** determines whether a stream declared in a taDec entry (pTaDec) is reservable or not.
  - It checks three aspects if it can be reserved or not
    - Check bandwidth
    - Check latency
    - Check resources (**99.8.5.17**)

- **Contribution in IEEE P802.1Qdd D0.7**
    - Suggestion 1: 99.8.5.17 checkResources could be made RA class specific
    - Suggestion 2: One or all of the 3 independent checks bandwidth, latency, resources could be made RA class specific.