Some Necessary Managed Objects in 802.1Qdd
Use Cases from Management and Configuration

Alexej Grigorjew
alexej.grigorjew@uni-wuerzburg.de
**Table 12-2—RAP Propagator Base Table**

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Operations supported(^a)</th>
<th>Conformance(^b)</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>taReg</td>
<td>Array of TalkerAnnounceRegistration</td>
<td>R</td>
<td>B</td>
<td>99.8.4.1.1, 99.8.4.1.2</td>
</tr>
<tr>
<td>taDec</td>
<td>Array of TalkerAnnounceDeclaration</td>
<td>R</td>
<td>B</td>
<td>99.8.4.2.1, 99.8.4.2.2</td>
</tr>
<tr>
<td>laReg</td>
<td>Array of ListenerAttachRegistration</td>
<td>R</td>
<td>B</td>
<td>99.8.4.3.1, 99.8.4.3.2</td>
</tr>
<tr>
<td>laDec</td>
<td>Array of ListenerAttachDeclaration</td>
<td>R</td>
<td>B</td>
<td>99.8.4.4.1, 99.8.4.4.2</td>
</tr>
<tr>
<td>localRaclass</td>
<td>Array of LocalRaClass</td>
<td>RW</td>
<td>B</td>
<td>99.8.4.5.1, 99.8.4.5.2</td>
</tr>
<tr>
<td>neighborRaclass</td>
<td>Array of NeighborRaClass</td>
<td>R</td>
<td>B</td>
<td>99.8.4.6.1, 99.8.4.6.2</td>
</tr>
<tr>
<td>maxProcessingDelay</td>
<td>integer</td>
<td>R</td>
<td>B</td>
<td>99.8.4.9</td>
</tr>
<tr>
<td>minProcessingDelay</td>
<td>integer</td>
<td>R</td>
<td>B</td>
<td>99.8.4.10</td>
</tr>
</tbody>
</table>

**RAP Draft D0.6 Comment Disposition**

**CI 12 SC 12.x.2 P29 L31 # 66**

**Specht, Johannes Self; Siemens AG**

**Comment Type T**

**Comment Status A**

Is it necessary to expose all the read-only parameters found in the pseudo-code (taReg, taDec, laReg, laDec, neighborRaClass, etc.) externally via Table 12-2 and subsequent tables?

The data types used in the pseudo-code is not necessarily how implementations have to be structured internally (in fact, one can imagine implementations with other structures optimized for speed, memory or both), but exposing all may imply some limitations on implementers for such optimizations.

Note that, for example, Table 12-3 contains portTransmitRate - this parameter also exists in Std 802.1Q in 8.6.8.2, clause 34 etc. - but is not externally visible.

**SuggestedRemedy**

**DISCUSS**

**Response**

**Response Status C**

ACCEPT IN PRINCIPLE.

Debugging implementation is not a good reason to have managed objects.

Move all the managed objects defined in D0.6 that are not crucial to RAP operation to Annex Z.

Add an editor's note that presentation on how to perform diagnosis using management and which managed objects are needed is needed.
Purpose of this Talk

We argue that we need managed objects for per-stream reservation data

- ... what exactly do we need?
- ... what are the important use cases?
- ... what else can we do with that?

This is not an exhaustive list!

- ... there are certainly more managed objects required for other use cases

Discussion

- ... what exactly is the problem with exposing arrays of TAs and LAs?
IMPORTANT RAP MANAGED OBJECTS
(PER-CLASS, PER-STREAM)
Use of RAP "Per-Class" Parameters During Reservability Check

```c
checkReservability(pTaDec) {
    tRaClass = getLocalRaClass(pTaDec.attr.priority);
    tPort = pTaDec.portRef;

    // check bandwidth
    tRequiredBandwidth = ceil(pTaDec.attr.CommittedInformationRate / port[tPort].portTransmitRate * 1.6);
    tRemainingBandwidth = portRaClass[tPort, tRaClass.id].maxBandwidth - portRaClass[tPort, tRaClass.id].allocatedBandwidth;
    if (tRequiredBandwidth > tRemainingBandwidth) {
        // unreservable due to insufficient bandwidth (failurecode x)
        return x;
    }

    // check latency
    for (tObsvRaClass: localRaClass[*]) {
        if (tObsvRaClass.ritd == "00-80-C2-xx") { // see Table 51-3
            tWorstCaseHopLatency = computeWorstCaseLatencySP(tObsvRaClass, pTaDec);
        } else if (tObsvRaClass.ritd == "00-80-C2-yy") { // see Table 51-3
            tWorstCaseHopLatency = computeWorstCaseLatencyATS(tObsvRaClass, pTaDec);
        }
        tMaxHopLatency = portRaClass[tPort, tObsvRaClass.id].maxHopLatency;
        if (tWorstCaseHopLatency > tMaxHopLatency) {
            // unreservable due to exceeding max hop latency (failurecode y)
            return y;
        }
    }

    // check resources
}
```

### Table 12-46—RAP Propagator RA Class Port Table row elements

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Operations supporteda</th>
</tr>
</thead>
<tbody>
<tr>
<td>domainBoundaryStatus</td>
<td>Boolean</td>
<td>R</td>
</tr>
<tr>
<td>maxStreamFrameSize</td>
<td>integer</td>
<td>RW</td>
</tr>
<tr>
<td>minStreamFrameSize</td>
<td>integer</td>
<td>RW</td>
</tr>
<tr>
<td>maxBandwidth</td>
<td>integer</td>
<td>RW</td>
</tr>
<tr>
<td>allocatedBandwidth</td>
<td>integer</td>
<td>R</td>
</tr>
<tr>
<td>maxHopLatency</td>
<td>integer</td>
<td>RW</td>
</tr>
</tbody>
</table>

12.35.5 RA Class Port Table

51.8.5.16 checkReservability(pTaDec)
What Does “Per Stream” Information Look Like?

99.8.4.1.1 TalkerAnnounceRegistration data type

The TalkerAnnounceRegistration data type is a structure that consists of a collection of member variables representing a Talker Announce registration in the RAP Propagator, as follows:

a) attr: A Talker Announce attribute (99.5.3).
b) portRef: A portRef (99.7.4.1) value, indicating a Bridge Port on which the attribute contained in item a) is registered.
c) isValid: A Boolean value indicating whether this Talker Announce registration is valid (TRUE) or not (FALSE), as determined by the validateTaReg procedure (99.8.5.1).
d) ingressStatus: The ingress status of this Talker Announce registration, as determined by the processTaAnnounce procedure (99.8.5.2), taking one of the following enumerated values:
   1) TA_RECVM_FAIL: This Talker Announce registration contains a failure code generated by an upstream station.
   2) TA_INGRESS_SUCCESS: This Talker Announce registration contains no failure code and is not failed on ingress of this Bridge.
   3) TA_INGRESS_FAIL: This Talker Announce registration contains no failure code but failed on ingress of this Bridge with a failure code contained in ingressFailureCode [item c), below].
e) ingressFailureCode: A RAP failure code.

TalkerAnnounce:
- StreamId: "B7:77:19:07:B4:18:00:01"
- StreamRank: 1
- AccMaxLatency: 150000 ns
- AccMinLatency: 10512 ns
- DataFrameParams:
  - DestinationMacAddress: "01:00:5E:00:00:01"
  - Priority: 7
  - VID: 5
- TSpec:
  - MaxTransmittedFrameLength: 1000 Bytes
  - MinTransmittedFrameLength: 64 Bytes
  - CommittedBurstSize: 8160 bits
  - CommittedInformationRate: 220000 bits/s

99.5.3 Talker Announce attribute and TLV encoding

A Talker Announce attribute TLV encodes in the Value field a set of parameters, followed by a series of sub-TLVs, as illustrated in Figure 99-12.

<table>
<thead>
<tr>
<th>Octet</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>StreamId</td>
<td>1</td>
</tr>
<tr>
<td>StreamRank</td>
<td>9</td>
</tr>
<tr>
<td>AccumulatedMaximumLatency</td>
<td>10</td>
</tr>
<tr>
<td>AccumulatedMinimumLatency</td>
<td>14</td>
</tr>
<tr>
<td>Data Frame Parameters sub-TLV</td>
<td>18</td>
</tr>
<tr>
<td>Token Bucket TSpec sub-TLV or MSRP TSpec sub-TLV</td>
<td>29</td>
</tr>
<tr>
<td>0 or 1 Redundancy Control sub-TLV</td>
<td>variable</td>
</tr>
<tr>
<td>0 or 1 Failure Information sub-TLV</td>
<td>variable</td>
</tr>
<tr>
<td>0 or more Organizationaly Defined sub-TLVs</td>
<td>variable</td>
</tr>
</tbody>
</table>

Figure 99-12—Value of Talker Announce attribute TLV

<table>
<thead>
<tr>
<th>Octet</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DestinationMacAddress</td>
<td>1</td>
</tr>
<tr>
<td>Priority</td>
<td>7</td>
</tr>
<tr>
<td>Reserved</td>
<td>7</td>
</tr>
<tr>
<td>VID</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 99-13—Value of Data Frame Parameters sub-TLV

<table>
<thead>
<tr>
<th>Octet</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxTransmittedFrameLength</td>
<td>1</td>
</tr>
<tr>
<td>MinTransmittedFrameLength</td>
<td>3</td>
</tr>
<tr>
<td>CommittedInformationRate</td>
<td>5</td>
</tr>
<tr>
<td>CommittedBurstSize</td>
<td>13</td>
</tr>
</tbody>
</table>

Figure 99-14—Value of Token Bucket TSpec sub-TLV
WHY PER-STREAM MANAGED OBJECTS?
(TAREG, TADEC, LAREG, LADEC)
Background Information: Latency Configuration & Burstiness Cascades

maxHopLatency[7] = 250µs

Prio=7
Burst=120B
Rate=100kbit/s
accMaxD=750µs

Effective Burst: 130B

Stream

Effective Burst: 127B

Effective Burst: 133B

Effective Burst: 136B

Effective Burst: 139B

Effective Burst: 142B
When increasing the maxHopLatency of a switch...

- The locally available resource latency increases
- But at the same time, streams utilize more latency resources downstream (higher burstiness)
So What Is The Configuration Task, Exactly?

Configuration and optimization problem (which can be approached by an NMS)

- Assumptions: (simplifications for this example)
  - four external priorities (4, 5, 6, 7) are used for reserved traffic
  - and they are mapped 1:1 to internal traffic classes
- For each switch, each port, each traffic class, and each type of resource (latency, bandwidth, internal buffers, ...), a threshold must be configured
- These thresholds affect utilization in the network (number of accepted stream reservations)
- Remember: changes in one device reversely affect resource utilization of other devices
Optimal Choice of maxHopLatency

Approach towards the optimization problem
- NMS collects the topology info, the current configuration, and the current list of TAs/LAs from each device
- NMS optimizes the configuration based on a digital twin
  - It needs to know the TSpecs and paths of every stream for that
- NMS sends the optimized configuration back to the switches

```text
New Optimized Configuration

Current Configuration (per class)
All TAs/LAs (per stream)
Network Topology

Network Management System
computeWorstCaseLatency()
```

### Table 12-43—RAP Propagator Base Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Data type</th>
<th>Operations supported</th>
</tr>
</thead>
<tbody>
<tr>
<td>laReg</td>
<td>Array of TalkerAnnounceRegistration</td>
<td>R</td>
</tr>
<tr>
<td>laDec</td>
<td>Array of TalkerAnnounceDeclaration</td>
<td>R</td>
</tr>
<tr>
<td>laReg</td>
<td>Array of ListenerAttachRegistration</td>
<td>R</td>
</tr>
<tr>
<td>laDec</td>
<td>Array of ListenerAttachDeclaration</td>
<td>R</td>
</tr>
<tr>
<td>neighborRaClass</td>
<td>Array of NeighborRaClass</td>
<td>R</td>
</tr>
<tr>
<td>maxProcessingDelay</td>
<td>integer</td>
<td>R</td>
</tr>
<tr>
<td>minProcessingDelay</td>
<td>integer</td>
<td>R</td>
</tr>
</tbody>
</table>

12.35.2 RAP Propagator Base Table
WHAT ELSE CAN WE DO?
How Many Resources Have Been Reserved? How Many Are In Use?

- **TSpec**: Burst 500 B
  - Rate 100 kbit/s

- **Measurement**: Burst 370 B
  - Rate 89 kbit/s

- **TSpec from RAP managed object**

- **Measurements can be done by...**
  - External devices
  - Counters on switches
  - ...
  - (not subject of this presentation)
Are Local Thresholds Being Exceeded?

- TSpec from RAP managed object
- The NMS should be able to alert the operator when devices exceed their TSpec

- This can happen rather quickly
  - Misconfigured devices
  - Software burst/rate limits, and variance introduced afterwards by the OS / NIC
  - Unexpected high delay variance in the network
  - FRER
  - General errors in software or hardware

- Filters/Meters can limit the damage, but the operator must be informed to fix the underlying problem
### Visualizations

**Many different visualizations possible**

- **Resource Utilization**
  - Available bandwidth
  - Used bandwidth
  - Burstiness
  - Latency
  - ... per class, per resource type, per link

- **Reconstruct and visualize stream paths**
  - Via Listener-Attach managed objects (LA-Reg, LA-Dec)
  - Plot & plan redundant paths
  - Assess reliability of a particular scenario

- **Monitor dangling reservations**
  - Talker-Announce objects (TA-Reg, TA-Dec)
  - Accumulated max. latencies & range of TAs
  - Available vs. reserved streams
CONCLUSION
Conclusion

Summary

▪ We need per-stream reservation data (TA-Dec, TA-Reg, LA-Dec, LA-Reg)
▪ ... for proper optimization of network utilization
▪ ... for monitoring of network state

By the way...

▪ MSRP already has per-stream managed objects

Discussion

▪ What exactly is the problem with exposing arrays of TAs and LAs?
▪ Is it the amount of information in general?
▪ Is it the specification of the data types?
▪ Are there any ways to circumvent the caveats?
THANK YOU!

Questions, comments, suggestions?

Alexej Grigorjew
University of Wuerzburg
Chair of Communication Networks
Email: alexej.grigorjew@uni-wuerzburg.de