UNIVERSITÄT DUISBURG ESSEN

Open-Minded

802.1Qcr Updates

Johannes Specht, University of Duisburg-Essen

General

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

Administrative

- Prepare a „skeleton“-draft for Atlanta in January 2017
  - Based on the last Qbv draft framemaker files
  - Structure and sections for the planned contents, find the appropriate places
  - Editor‘s notes describing outlining the contents
  - Integrate some existing figures

Technical

- Consecutive slides sketch the intended technical content planned right now
- Stay high level – the author believes that detailed technical discussions should be based on specific drafts of 802.1Qcr, complemented by future presentations for specific topics

Planned Technical Content

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

Model for 802.1Qcr

1. Basic model as described in May 2016, Budapest: http://www.ieee802.org/1/files/public/docs2016/Qcr-specht-specification-mapping-proposal-0516-v01.pdf
2. Additions based on feedback on this model from TSN members, which the author believes can already be incorporated
3. Additions in order to be in line with the PAR

Additions (Feedback and PAR)

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

1. **Shaper Algorithm Bug**
   There was an arithmetic bug in the pseudo code of the shaper. Thanks for pointing it out and fixing it!
2. **Priority Mapping and Re-Mapping**
   ATS streams can vary their priority at different ports along their paths. The model show in Budapest introduced a new mapping mechanism. However, it seems IPV (cmp. 802.1Qci and 802.1Qch) provides a majority of what is needed, as opposed contrast to PCP/DEI in 802.1Q tags.
3. **Delay Calculation (Informative Annex)**
   „The project provides an informative framework for worst case delay analysis in static networks/configurations."
   (cmp. https://development.standards.ieee.org/P946200033/par, sec. 5.2.b.)

Additions (Feedback and PAR)

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

4. **Model vs. Implementation (Informative Annex)**
   The presented model has the intention to re-use as much as possible from what is already 802.1Q, including contents from 802.1Qci. As a consequence, the essential shaper arithmetic is at ingress near Qci meters, the queueing is at egress.
   However, Implementers are free to implement this model:
   - Directly in this distributed manner,
   - entirely on egress (cmp. http://www.ieee802.org/1/files/public/docs2015/new-tsn-specht-ubs-queues-0521-v0.pdf),
   - etc.

   We can have an informative annex to describe potential differences, while the normative parts use the proposed model as long as interoperability is ensured.

Thank you for your Attention!

Questions, Opinions, Ideas?

Johannes Specht
Dipl.-Inform. (FH)

Dependability of Computing Systems
Institute for Computer Science and
Business Information Systems (ICB)
Faculty of Economics and
Business Administration
University of Duisburg-Essen

Schuetzenbahn 70
Room SH 502
45127 Essen
GERMANY
T +49 (0)201 183-3914
F +49 (0)201 183-4573

Johannes.Specht@uni-due.de
http://dc.uni-due.de

Updated Shaper FSM Algorithm

Shaper FSM: Per Frame Processing

UNIVERSITÄT DUISBURG ESSEN
Open-Minded

Parameters

- *Committed Information Rate (CIR) [bit/s]*
  The (constant) data rate of the token bucket
- *Committed Burst Size (CBS) [bit]*
  The capacity of the token bucket

State

- *Bucket Empty Time [time]*
  The time at which the token bucket was empty, initialized to "-inf"
- Bucket level storage not needed

Error Handling

- On exceeded Maximum Residence Time
- On exceeded Frame Length … Qci does already provide this on per stream level[1]
  → can be skipped here if applicable

```
void processFrame(Frame frame, RxPriority rxPriority, Shaper shaper) {
  time dLengthRecover  = frame.length /
                         shaper.param.committedInformationRate;
  time dEmptyToFull    = shaper.param.committedBurstSize /
                         shaper.param.committedInformationRate;
  time tShaperEligible = shaper.state.tBucketEmpty + dLengthRecover;
  time tBucketFull     = shaper.state.tBucketEmpty + dEmptyToFull;
  boolean frameValid   = true;

  frame.tEligible = max( frame.tArrival,
                         rxPriority.state.tEligible,
                         tShaperEligible);

  frameValid &= frame.tEligible <= frame.tArrival +
                                   rxPriority.param.dResidenceMax;
  frameValid &= frame.length    <= shaper.param.lengthLimit;

  if (frameValid){
    // Normal: Frame passes and state is updated
    rxPriority.state.tEligible = frame.tEligible;
    shaper.state.tBucketEmpty = (frame.tEligible < tBucketFull) ?
           dLengthRecover + shaper.state.tBucketEmpty :
           dLengthRecover + frame.tEligible – dEmptyToFull;
} else {
    // Error: Drop frame and trigger further reaction (blocking, etc.)
  }
}
```

1: 802.1Qci, 8.6.5.1, item e)1)