ATS Updates

Johannes Specht, University of Duisburg-Essen

Purpose

Content

1. Discuss a model of ATS close to 802.1Q
2. Show relationships to a 3-port bridge supporting ATS with a single priority level
3. Improve the algorithm (the backgrounds will be explained)
4. Give a clear overview of the parameters and variables, their locations in the proposed model, etc.
5. Address some relationships with 802.1Qci

Properties of the model

- Re-use what is already there in 802.1Q, if appropriate
- Essentially the same performance characteristics as the ATS model already shown
- More freedom for implementers/architecture specific simplifications (assumed, but let‘s discuss this…)

Proposed Specification Model

Differences - Outline

“priority” instead of “sub-priority”

- transmission priority levels
- reception priority levels

Shaper Finite-State Machines (FSMs) associated to transmission ports, not reception ports

- In the specification model (but no implementation requirement)
- Relax requirements for shaper FSMs, enables operation at packet rate (input) instead of *(portcount-1)*packet* rate (output)
- Less FSM state variables in case of multicast
- Close to Qci meters, which are, … FSMs

Change the Algorithm to a timestamp-based Token-Bucket Algorithm

- Shaper FSMs can be shared by stream aggregates (shared FSM state among streams)
  - Whenever the packets of a stream aggregate will stay in FIFO order to the listener
  - potentially in other cases (under investigation)
- More compatibility for different traffic types (including bursty streams), no delay penalties for stream aggregates with low bandwidth
- Side effect: We can get lower guaranteed delays (faster)

Reception Port - Associated Parts

Timing

- All timestamps in this slide set derived from a local system time in a bridge → independent of gPTP (or any other global clock synchronization)

Shaper FSMs

- Compute frame output timing of associated shaped queues in transmission ports.
- Mark frames with *eligibility times* (meta-value in frames), non-decreasing for each reception priority, at which these frames leave the shaped queues

Reception Priority

- Shaper FSMs of one reception priority use a second *eligibility time,* shared by all FSMs of the reception priority:

  Stores the time when the last received frame will leave the associated shaped queues in the transmission ports.

- Additional *Max. Residence Time* Parameter per reception priority:

  Limits frame residence times in associated shaped queues (babbling Idiot handling).

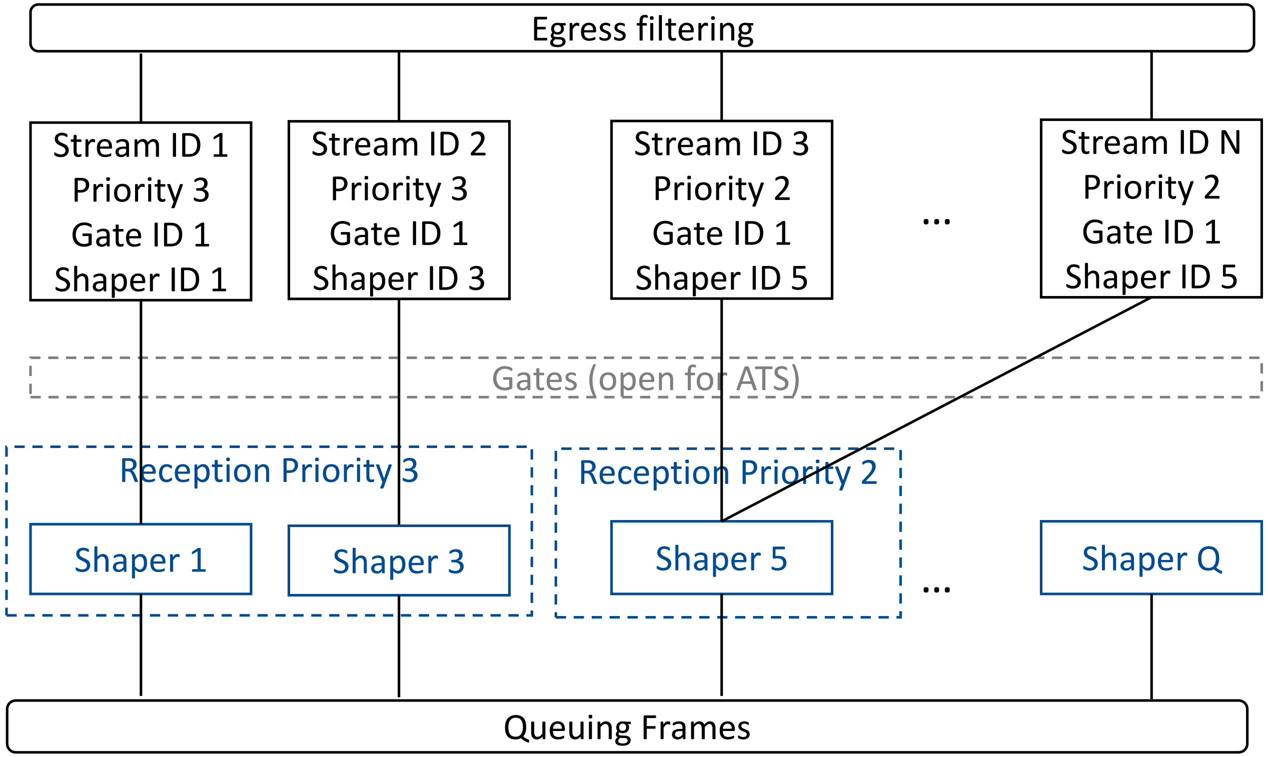
→Table per reception Port: *Reception priority table*

Model vs. Implementations

Close to metering (sec. 8.6.5), but

- Meters operates *per* reception port[1], but not necessarily *implemented in* reception ports
- Also applies for Shaper FSMs and, on transmission side, ATS subdivides shaped queues per reception port

→ Does not really matter for ATS where the FSMs are is implemented



Reception Priority	*Max. Residence Time (parameter)*	*Eligibility Time (state)*
2	5000	128742
3	10000	342613
…	…	…

[1] 802.1Q-2014, 8.6.5, p.125, Note 2

Transmission Port – Associated Parts

Transmission Priority Assignment

- *Transmission Priority Table*, only used by ATS … yet
- Maps frames from shaper FSMs to 802.1Q (Traffic Class -)Queues

New Traffic Selection Algorithm

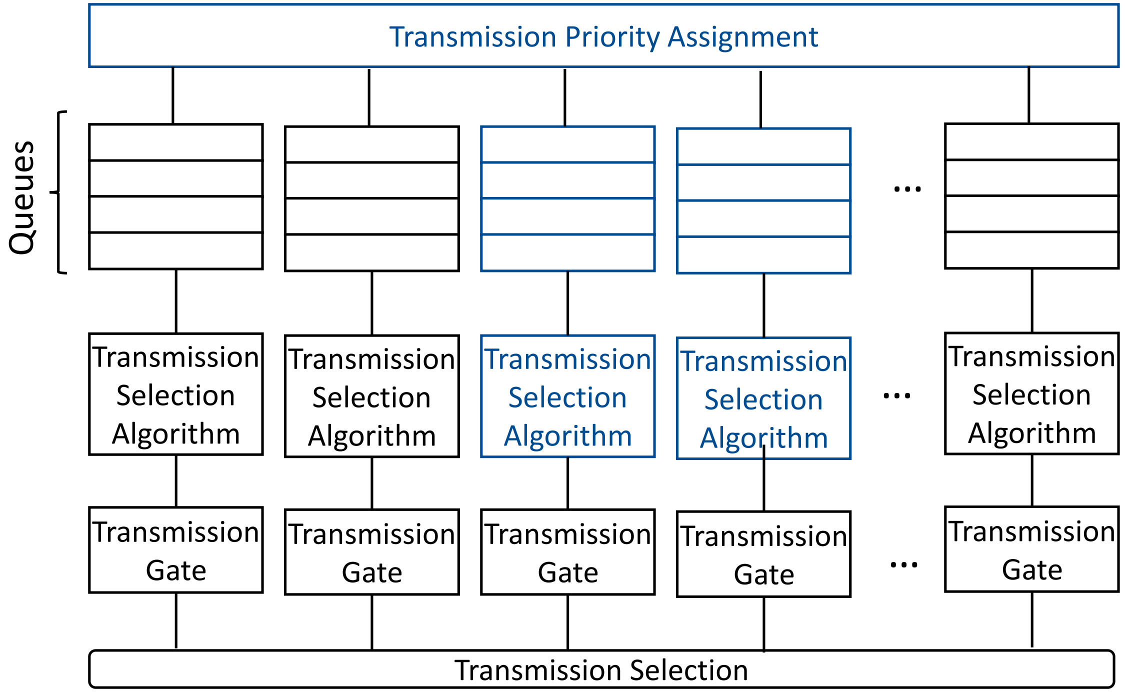
1. Only frames with eligibility times greater than "now" are selected
2. Frames are selected in ascending order of their eligibility times
3. Retain order of frames from the same Shaper FSM with equal eligibility times

Queues used by ATS

- Only frames with eligibility time
- Not a single FIFO structure – which is nothing new[1]
- The trick here are non-decreasing eligibility times per reception priority:

  → Can be implemented like outlined in http://www.ieee802.org/1/files/public/docs2015/new-tsn-specht-ubs-queues-0521-v0.pdf …

  → … or completely different


Transmission Priority Assignment
Queues
…
Transmission Selection Algorithm
Transmission Selection Algorithm
Transmission Selection Algorithm
Transmission Selection Algorithm
…
Transmission Selection Algorithm
Transmission Gate
Transmission Gate
Transmission Gate
Transmission Gate
…
Transmission Gate
Transmission Selection

Shaper ID	Transmission Priority (Parameter)
1	4
3	3
…	…

[1] 802.1Q-2014, sec. 8.6.6, NOTE 3

Shaper FSM: Per Frame Processing

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) ?
                               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)

Looking at a small Bridge …

Looking at a small Bridge

Possible Implementation Range [1]

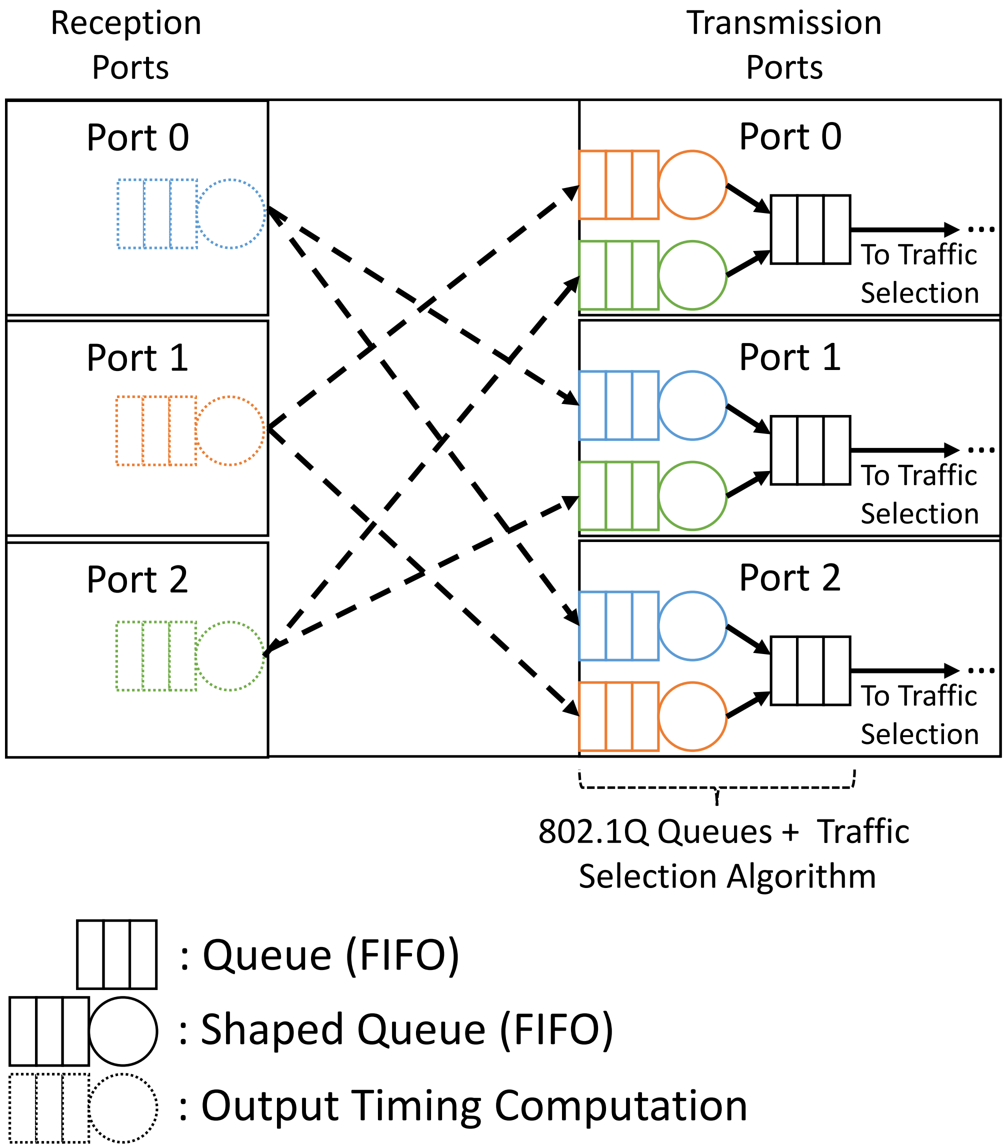
- Output timing computations in reception ports
  - Shaper FSMs located in reception ports
  - Per reception priority tracked
  - Eligibility times contained in frames

…

- Output timing computations in transmission ports
  - Shaper FSMs located at the output of shaped queues in transmission ports
  - Per reception priority eligibility time not needed (time is non-decreasing anyway)
  - Eligibility times not contained in frames

Impact

- In reception ports:
  - Less state variables
  - FSMs executed once per packet (line speed)
- In transmission ports:
  - Higher level of aggregated streams per FSM possible if multicast streams are present



[1] Cmp. http://www.ieee802.org/1/files/public/docs2015/new-tsn-specht-ubs-queues-0521-v0.pdf

Issues, Questions & Observations

Precision

Length-Rate-Quotient

```
time dLengthRecover = frame.length/shaper.param.committedInformationRate;
```

- Looks like a division, but implemented as multiplication
- Specification needed for rounding error:
  - Not necessarily accurate (limited bits) but devices should make the same rounding errors
  - Rounding errors should not accumulate over time (e.g., add the remainder from the last multiply-accumulate operation `shaper.state.tBucketEmpty + dLengthRecover`)

Time Types and Time Source

- Preferably close to the MII
- Precision is good, but power-of-two granularities should be ok in general (1,2,4,8,… octet times) – required time range is limited
- Easiest way to handle oscillator deviations (e.g., +-100 ppm) is slight over-reservation among adjacent shaper FSMs along the path. Should be ok…:

$$\left(\frac{10^6+100}{10^6-100}\right)^{Hops}$$

#Hops	Over-reserved bandwidth
1	~0.02 %
10	~0.2 %
100	~2 %

Priority Number vs. Priority Number vs. …

Reception Priority – IPV not used…

- For ATS, I believe a little number of priority levels is sufficient – 8 already appears large
- IPV allows more flexibility …
- but I am unsure what are the implications of both (Tags, other limitations, etc.)?

Transmission Priority Table

- This table permits per hop priority re-assignments, which
  1. allows a class based (network-global) scheme, but …
  2. … does not limit to it, and thus permits heavily engineered priority assignment on per hop granularity
- Even for the heavily engineered case, it could also be associated to reception ports … for unicast streams, **<u>but</u>**
  …for multicast streams, this would prohibit different transmission priorities at different transmission ports
  →Locating the table in reception ports would be a limitation here
- Moreover, assignment is based on Shaper ID, not Stream ID … which keeps the table smaller because Shaper FSMs can be shared by stream aggregates
- Moreover, unique Shaper IDs among multiple ports are required for unambiguous mapping
- I am unsure how hard the proposed location on the transmission side is to realize (required 802.1Q changes, implementation/efficient encoding, etc.)?

Stream ID	Shaper ID
1	1
2	3
…	…

Likely already in 802.1Qci (there seems to be a mutual exclusion between Meters and Shapers, be see later slide)

Shaper ID	Transmission Priority (Parameter)
1	4
3	3
…	…

Transmission Port

Shapers vs. Meters

Recycling Parameters

- Matching 802.1Qci Parameters, in particular[1]:
    - ieee8021PSFPFlowMeterCIR [exists]
    - ieee8021PSFPFlowMeterCBS [exists]
    - ieee8021PSFPFlowMeterResidenceTimeMax [new]
- Issues:
    - Naming tied to Per Stream Filtering and Policing (*PSFP*) and *FlowMeter*
    - Parameter CBS in Qci is octets, in ATS it is bits

Recycling Logic

- Shaping can be considered a stronger form of metering, i.e., Shaped Traffic could live without additional Metering…
- If Strict Priority, Credit-Based Shaper, Qbv or Qch is used, ATS is not used (XOR)
- The shown pseudo-code could be extended to unify precisely the operation of timestamp-based meters with high accuracy (See next slide)

[1] Cmp. 802.1Qci-D1.1, tables 12-31 and 17-30

Single Bucket Metering Code[1]

Shaper FSM

```
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) ?
                                   shaper.state.tBucketEmpty + dLengthRecover :
                                   frame.tEligible - dEmptyToFull;
  } else {
    // Error: Drop frame and trigger further reaction (blocking, etc.)
  }
}
```

Meter FSM

```
void processFrame(Frame frame, Meter meter) {
  time dLengthRecover   = frame.length /
                          meter.param.committedInformationRate;
  time dEmptyToFull     = meter.param.committedBurstSize /
                          meter.param.committedInformationRate;
  time tMeterEligible   = meter.state.tBucketEmpty + dLengthRecover;
  time tBucketFull      = meter.state.tBucketEmpty + dEmptyToFull;
  boolean frameValid    = true;






  frameValid &= tMeterEligible  <= frame.tArrival;

  frameValid &= frame.length    <= meter.param.lengthLimit;

  if (frameValid){
    // Normal: Frame passes and state is updated

    meter.state.tBucketEmpty  = (tMeterEligible < tBucketFull) ?
                                   meter.state.tBucketEmpty + dLengthRecover :
                                   tMeterEligible - dEmptyToFull;
  } else {
    // Error: Drop frame and trigger further reaction (blocking, etc.)
  }
}
```

[1] Easiest form of metering for illustration, could be extended

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