Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Institute of Computer Science
Chair of Communication Networks
Prof. Dr. Tobias Hoßfeld

Traffic Specification Types in Qdd

IEEE 802.1 TSN Plenary Meeting – July 16, 2021

Alexej Grigorjew
alexej.grigorjew@uni-wuerzburg.de

Feng Chen
chen.feng@siemens.com

Johannes Specht
johannes.specht.standards@gmail.com

Why Should We Look Into TSpecs?

802.1Qdd Resource Allocation Protocol

- Protocol and traffic models for distributed bridge-local reservation
- Allocation of resources (bandwidth, queues, **latency**, …) based on traffic volume and priority

Traffic Volume → Traffic Specification

- What kind of information do we need exactly?
- How can that information be communicated?
- How can that information be represented internally by the switches?

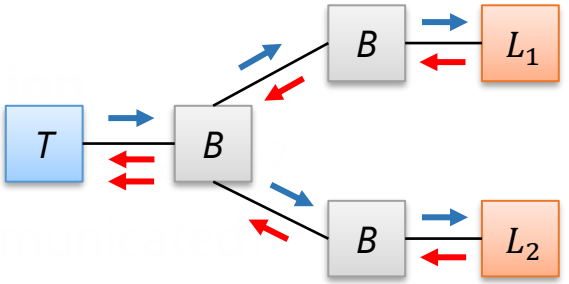
After a closer look…

- What if the information that we need is missing?
- How to be (backwards) compatible with MSRP?
- How to support the various different kinds of shapers?
- How can we deal with inaccuracies, such as clock drifts?

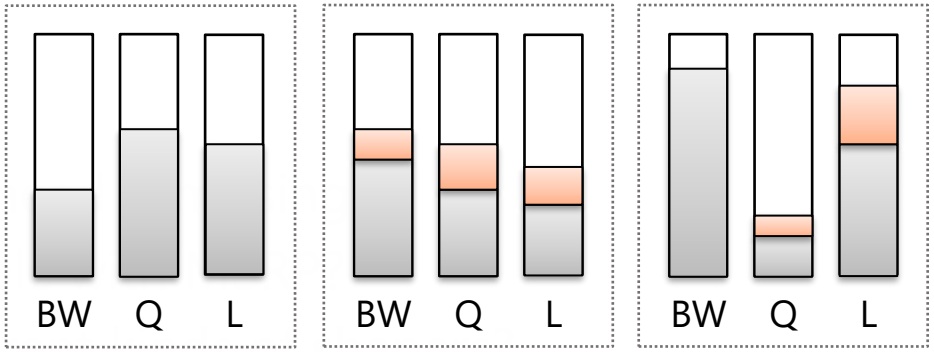
Why Should We Look Into TSpecs?

802.1Qdd Resource Allocation Protocol

- Protocol and traffic models for distributed bridge-local reservation
- Allocation of resources (bandwidth, queues, **latency**, …) based on traffic volume and priority



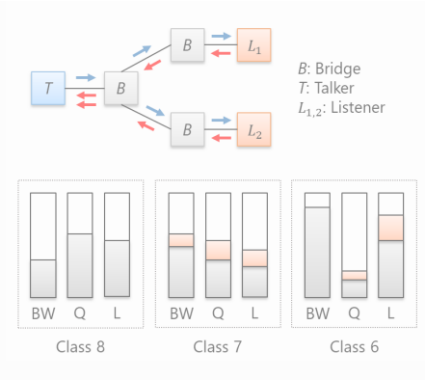
B: Bridge
T: Talker
$L_{1,2}$: Listener



Why Should We Look Into TSpecs?

802.1Qdd Resource Allocation Protocol

- Protocol and traffic models for distributed bridge-local reservation
- Allocation of resources (bandwidth, queues, **latency**, …) based on traffic volume and priority



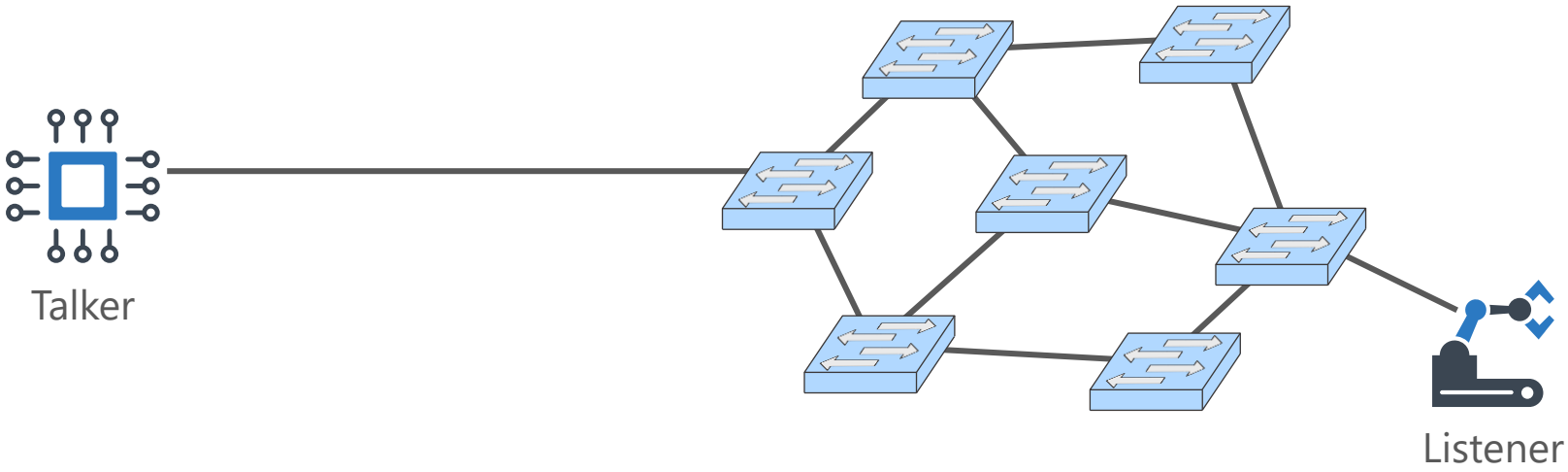
Traffic Volume → Traffic Specification

- What kind of information do we need exactly?
- How can that information be communicated?
- How can that information be represented internally by the switches?

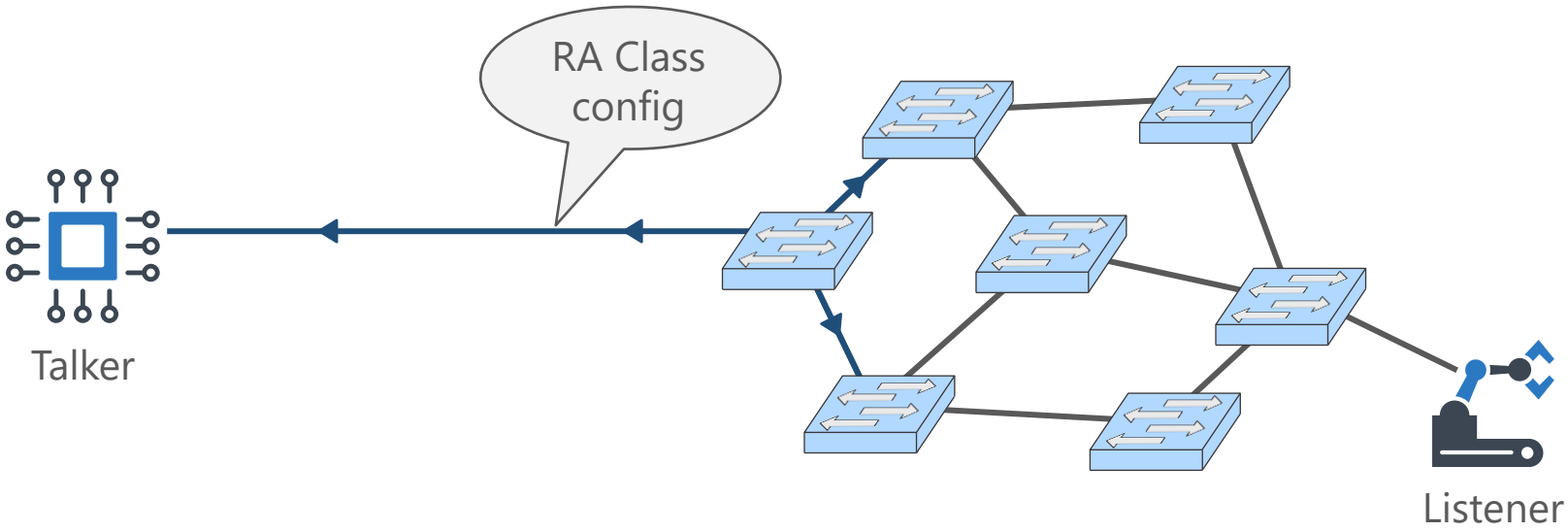
After a closer look…

- What if the information that we need is missing?
- How to be (backwards) compatible with MSRP?
- How to support the various different kinds of shapers?
- How can we deal with inaccuracies, such as clock drifts?

General Reservation Process (simplified)



General Reservation Process (simplified)

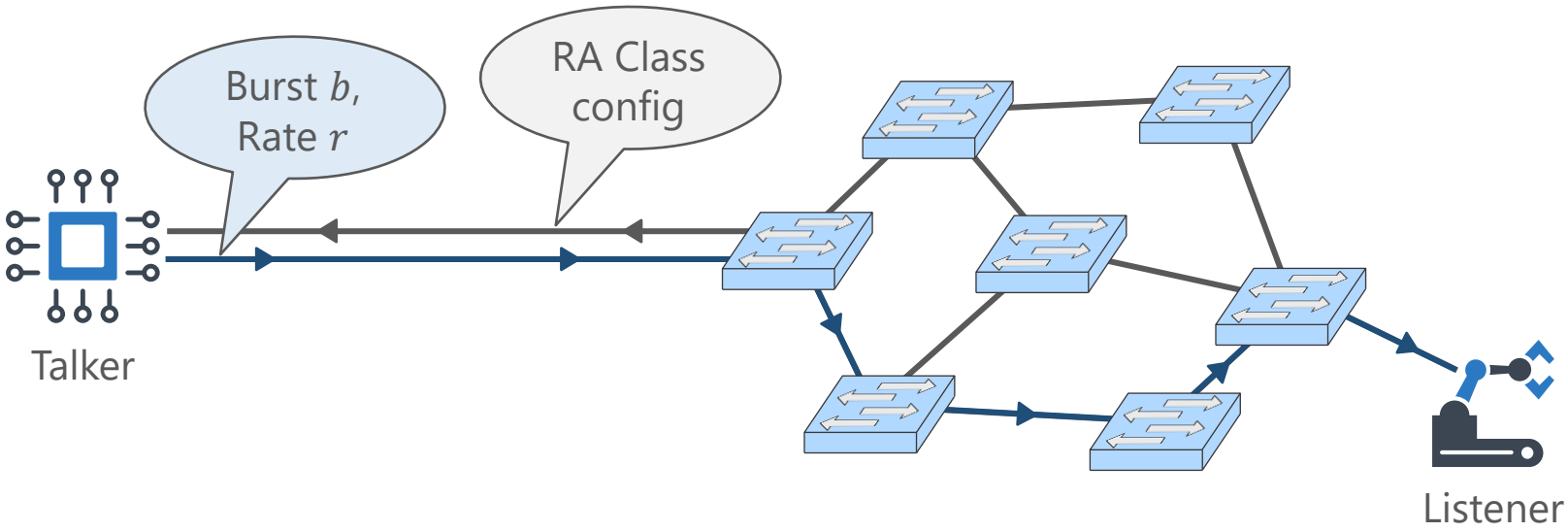


- RA class configurations (supported classes, shapers, …)

cf. RA Class Template:

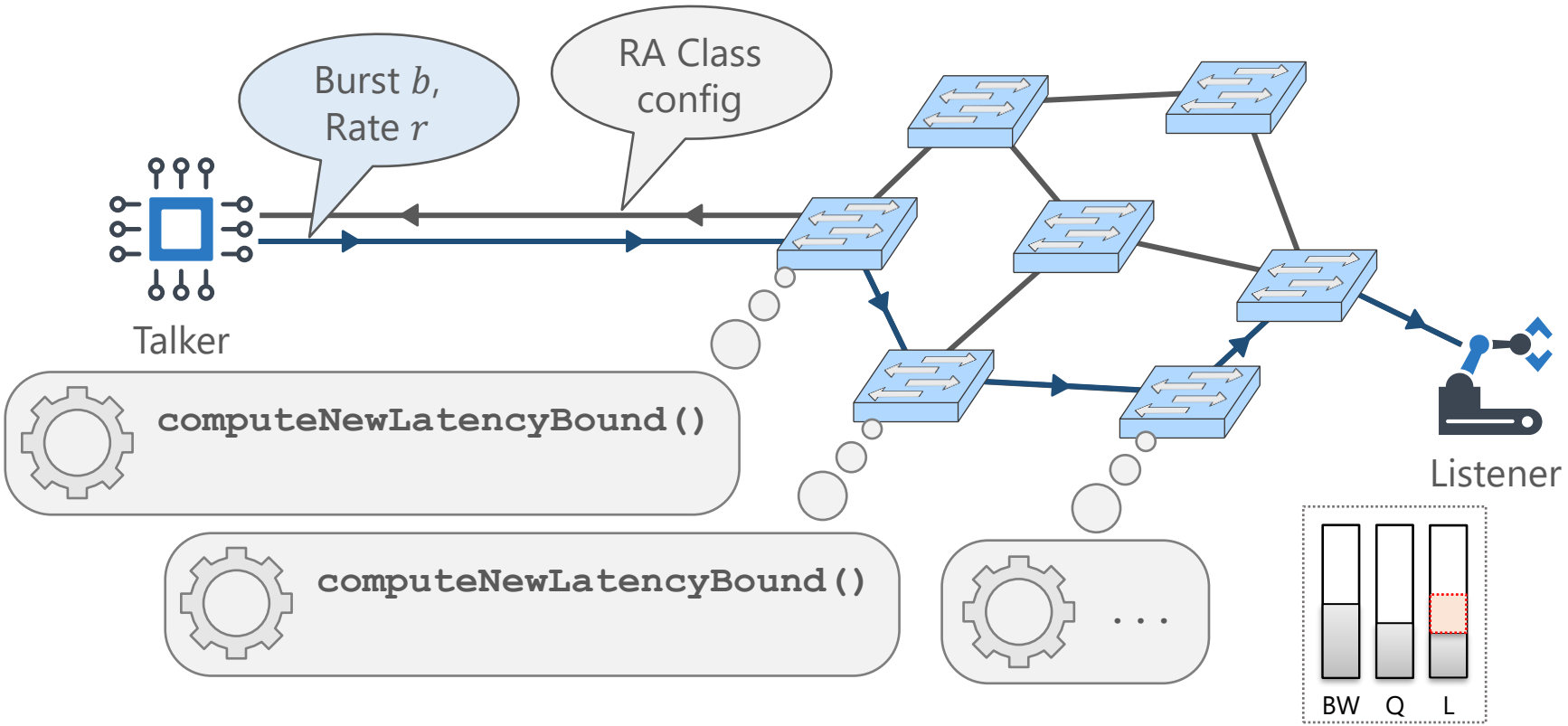
https://www.ieee802.org/1/files/public/docs2021/dd-chen-rap-introduction-0521-v01.pdf

General Reservation Process (simplified)



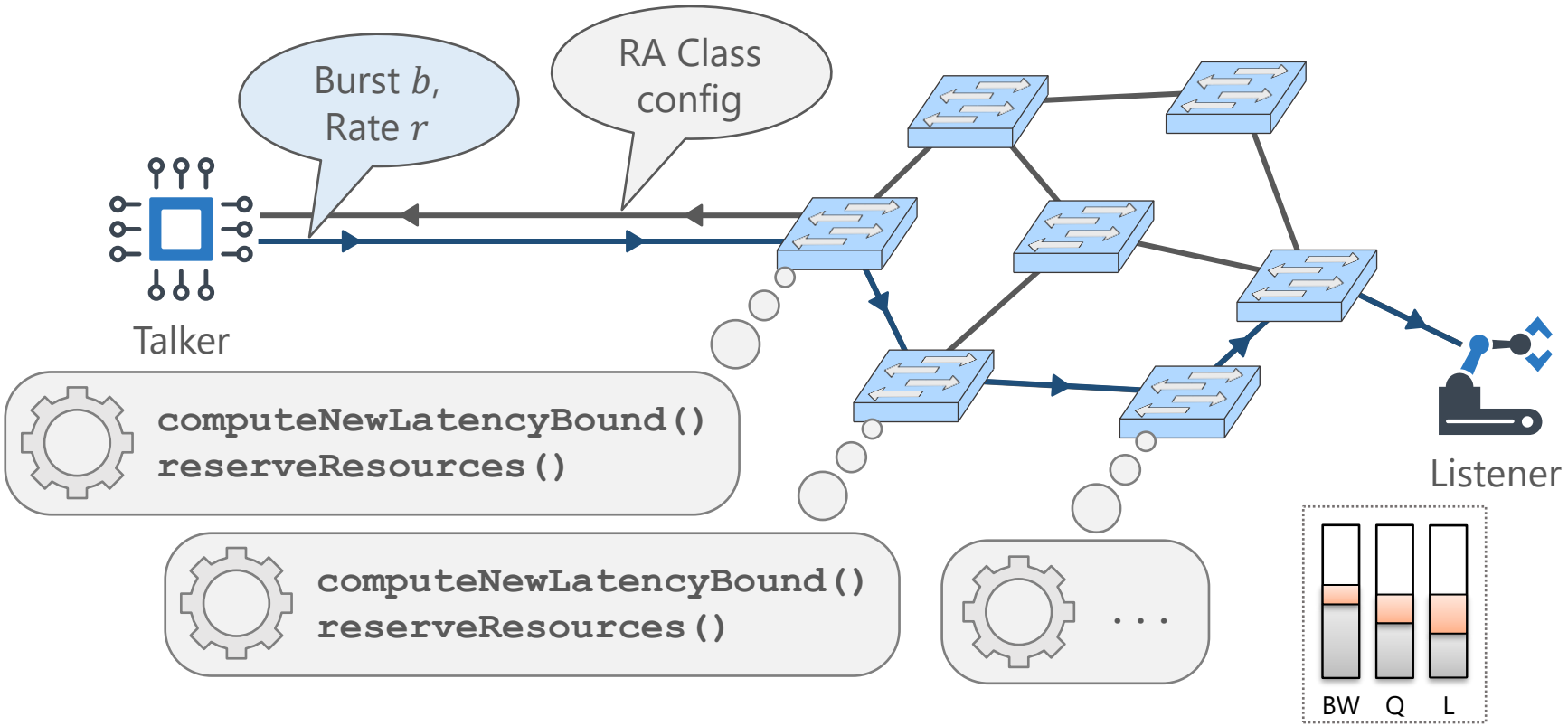
- RA class configurations (supported classes, shapers, …)
- Traffic specification

General Reservation Process (simplified)



- RA class configurations (supported classes, shapers, ...)
- Traffic specification
- Worst-case delay computation (based on shaper)

General Reservation Process (simplified)



- RA class configurations (supported classes, shapers, ...)
- Traffic specification
- Worst-case delay computation (based on shaper)
- Listener subscription, reserve resources, ...

What about the MSRP TSpec?

MSRP TSpec
- CMI Duration
- Max CMI Frames
- Max Frame Size

- The MSRP TSpec **could** be used, but it's not perfect
- What kind of information do we need exactly?
- From IEEE Std 802.1Qcc, Section 35 (SRP):

IEEE Std 802.1Qcc-2018
IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—
Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements

Figure 35-11 specifies the encoding of the value for the TrafficSpecification TLV.

	Octet	Length
Interval	1	4
MaxFramesPerInterval	5	2
MaxFrameSize	7	2
TransmissionSelection	9	1

Figure 35-11—Value of TrafficSpecification TLV

Figure 35-12 specifies the encoding of the value for the TSpecTimeAware TLV. The presence of the optional TSpecTimeAware TLV is handled as specified in 46.2.3.5 for the presence of the TSpecTimeAware group.

	Octet	Length
EarliestTransmitOffset	1	4
LatestTransmitOffset	5	4
Jitter	9	4

Figure 35-12—Value of TSpecTimeAware TLV

What about the MSRP TSpec?

- The MSRP TSpec **could** be used, but it's not perfect
- What kind of information do we need exactly?
- Latency model, IEEE Std 802.1Qcr (ATS), Annex V:

MSRP TSpec
- CMI Duration
- Max CMI Frames
- Max Frame Size

Asynchronous Traffic Shaping
- Burst Size
- Data Rate
- Max Frame Size
- Min Frame Size

$$d_{BU,max}(n,f) = \frac{\sum_{g \in F_H(n,f) \cup F_S(n,f)} b_{max}(n,g) - l_{min}(f) + l_{LP,max}(n,f)}{R(n) - \sum_{g \in F_H(n,f)} r_{max}(n,g)} + \frac{l_{min}(f)}{R(n)}$$

where

$F_H(k,f)$ and $F_H(k,h)$	denote the set of streams transmitted in a numerically higher traffic class (8.6.8) than stream f and a stream h, respectively, at the upstream transmission Port of the kth hop
$F_S(k,f)$ and $F_S(k,h)$	denote the set of streams transmitted in the same traffic class as stream f, including stream f and stream h, respectively, at the upstream transmission Port of the kth hop
$l_{LP,max}(k,f)$ and $l_{LP,max}(k,h)$	denote the maximum interference length, in bits, by any numerically lower traffic class than the class of stream f and a stream h, respectively, at the upstream transmission Port of the kth hop
$l_{min}(f)$ and $l_{min}(h)$	denote the minimum frame length of stream f and a stream h, respectively, in bits, including all media-dependent overhead (8.6.11.3.11, 12.4.2.2)
$b_{max}(k,g)$	is the maximum burst size associated with a stream g at the kth hop, in bits
$r_{max}(k,g)$	is the committed information rate of stream g at in the upstream device of the kth hop, in bits per second
$R(k)$	is the transmission rate, in bits per second, that the underlying MAC Service that supports transmission through the upstream transmission Port of the kth hop provides

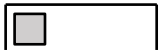
What about the MSRP TSpec?

MSRP TSpec
- CMI Duration
- Max CMI Frames
- Max Frame Size

- The MSRP TSpec **could** be used, but it's not perfect
- What kind of information do we need exactly?
- MSRP TSpec is inefficient for many applications
  - Burst size and data rate must be calculated
  - Min frame size is missing → default 64 bytes
  - Overestimation when CMI is small

Asynchronous Traffic Shaping
- Burst Size
- Data Rate
- Max Frame Size
- Min Frame Size

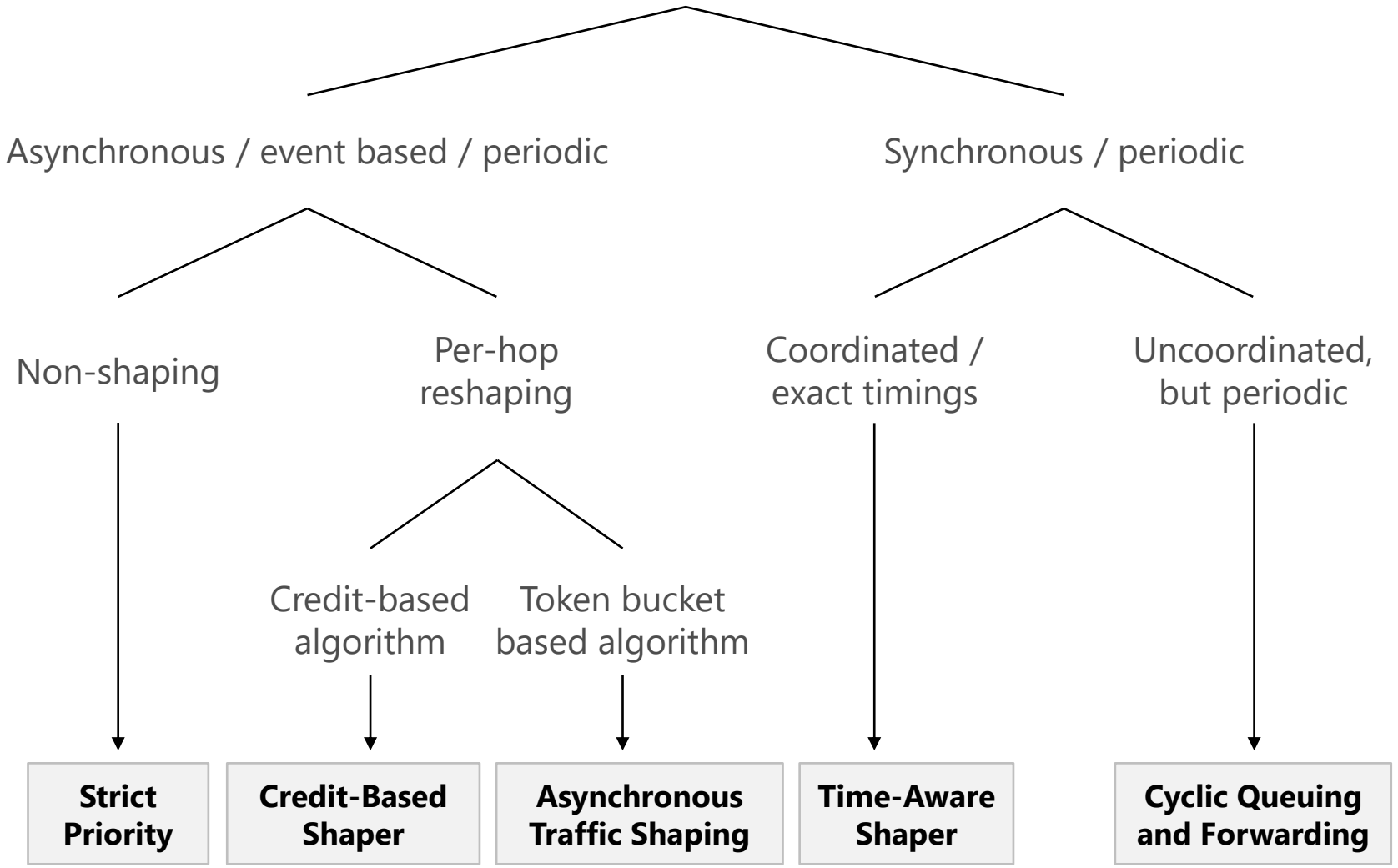
Transmission: 1 burst every 1 ms



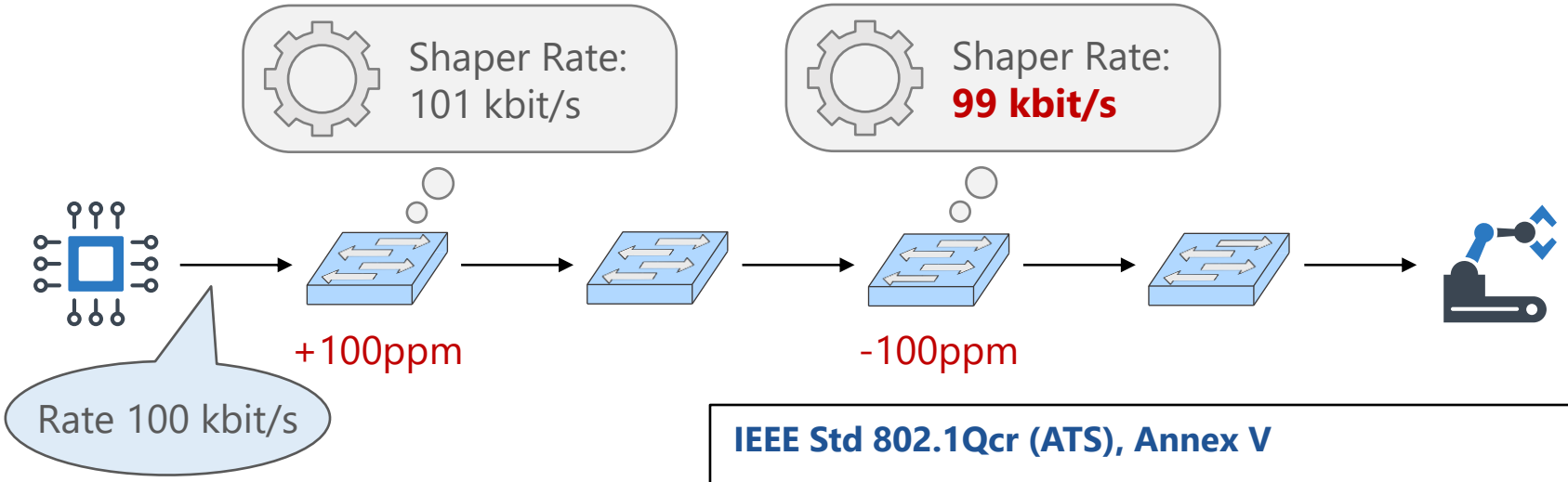
CMI 250 µs → calculated data rate is 400% of real rate

- MSRP TSpec not ideal for some transmission behaviors and shapers
  - Asynchronous event-based transmissions (token-bucket shaped)
  - **Simple** periodic transmission behavior (→ CQF)
- MSRP TSpec only carries Talker information
  - Little per-hop adjustments possible
  - E.g. no support of clock drift adjustments → per-hop TSpec fields

Brief Shaper Overview



Clock Drift TSpec Adjustments



IEEE Std 802.1Qcr (ATS), Annex V

V.8 Inter-device clock rate deviations

The clock constraints in 8.6.11 limit ATS scheduler clock instances and transmission selection clock instances in a Bridge to effectively operate at the same rate, although differences within [ClockOffsetMin,ClockOffsetMax] are permitted (8.6.11.2). As a basic property of asynchronous mechanisms such as ATS, no such limitation exists between different devices (i.e., clocks in different devices are not synchronized).

Deviations of clocks from their nominal rates (e.g., within oscillator tolerances) affect the spacing between successive frames according to the assigned eligibility times (8.6.11.3.2, 8.6.8.5). If the upstream device at a hop runs faster than nominal (e.g., +100 ppm), and a connected downstream Bridge at this hop runs slower than nominal (e.g., –100 ppm), the backlog as well as the per hop delay in the downstream Bridge could grow under peak load conditions.

The situation can be prevented by management constraints (12.31.5) on $r_{max}(k, g)$ for $1 < k < n$, such that Equation (V-5) holds for any stream g.

$$r_{max}(k, g) \geq \frac{1 + 10^{-6}\Delta_{CR,\,max}(k-1)}{1 - 10^{-6}\Delta_{CR,\,max}(k)} r_{max}(k-1, g) \qquad \text{(V-5)}$$

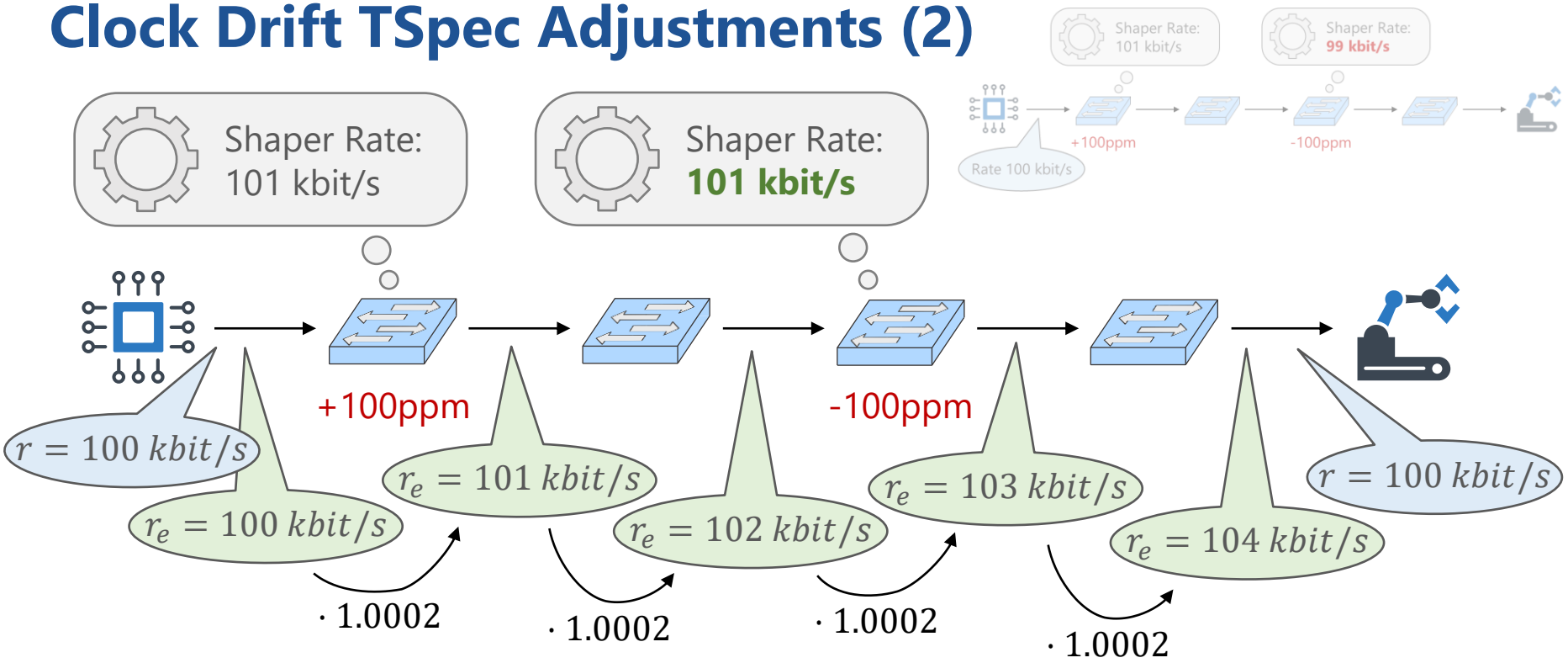
where

$\Delta_{CR,\,max}(k)$ is the upper bound over all absolute rate deviations of all ATS scheduler clocks and transmission selection clocks from their nominal rate in the upstream device of the kth hop available via the associated ClockRateDeviationMax parameter (12.31.8.4), in ppm (e.g., $\Delta_{CR,\,max}(k) = 100$)

$r_{max}(1, g)$ is the committed information rate of stream g at the Talker station (49.2)

+ 0.02%

Clock Drift TSpec Adjustments (2)



Suggestion: More Fields

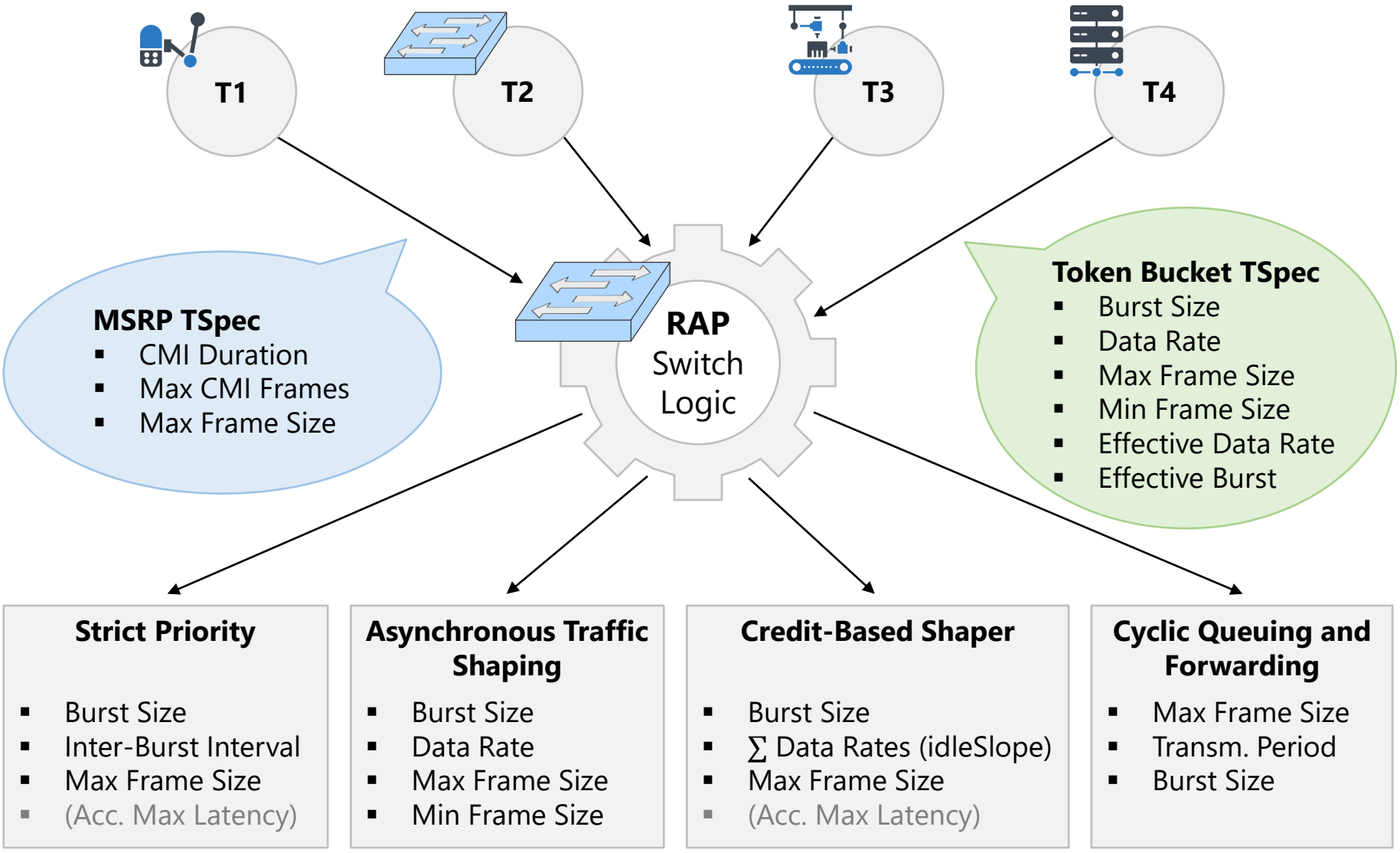
- End-to-end Talker TSpec fields
- Effective per-hop TSpec fields
- Provide extra fields for per-hop adjustments
- To combat inaccuracies (such as clock drift)
- To facilitate transitions (e.g., ATS → SP)
  (not subject of this contribution!)

Token Bucket TSpec

- Burst Size
- Data Rate r
- Max Frame Size
- Min Frame Size

- Effective Data Rate r_e
- Effective Burst

Perspectives: Available vs. Required Information



(required information for different worst-case latency algorithms)

ATS Latency Math Example

```
int computeWorstCaseATSLatency(...) {
  // ... TSpec.getEffectiveBurst() ...
  // ... TSpec.getEffectiveDataRate() ...
  // ... TSpec.getMaxFrameSize() ...
  // ... TSpec.getMinFrameSize() ...
}
```

→ (cf. Qcr Annex V)

Asynchronous Traffic Shaping

- Burst Size
- Data Rate
- Max Frame Size
- Min Frame Size

```
class TrafficSpecification {
  // [...] fields

  int getMaxFrameSize() {
    if (maxFrameSize > -1) { return maxFrameSize; }
    else { throw new ReservationError("maxFrameSize not initialized"); }
  }
  int getMinFrameSize() {
    if (minFrameSize > -1) { return minFrameSize; }
    else { throw new ReservationError("minFrameSize not initialized"); }
  }
  int getBurst() {
    if (burst > -1) { return burst; }
    else { throw new ReservationError("burst not initialized"); }
  }
  int getDataRate() {
    if (dataRate > -1) { return dataRate; }
    else { throw new ReservationError("dataRate not initialized"); }
  }
  int getEffectiveBurst() {
    if (effectiveBurst > -1) { return effectiveBurst; }
    else { throw new ReservationError("effectiveBurst not initialized"); }
  }
  int getEffectiveDataRate() {
    if (effectiveDataRate > -1) { return effectiveDataRate; }
    else { throw new ReservationError("effectiveDataRate not initialized"); }
  }

  // [...] constructors
}
```

- Make **internal** data structures independent of TSpec type

  (no switch/case blocks in algorithms)

- Reservation is still based on different types!

- If data is not initialized: reservation fails; return the respective error code to Talker

Different Constructors for Different TSpecs

Populate the same internal fields based on different TSpec types:

```
class TrafficSpecification {
  // [...] fields

  // Token-Bucket TSpec
  public TrafficSpecification(int maxFS, int minFS, int b, int dr, ...) {
    maxFrameSize = maxFS;
    minFrameSize = minFS;
    burst = b;
    dataRate = dr;
    ...
  }

  // MSRP TSpec
  public TrafficSpecification(int cmiDuration, int maxCmiFrames, int maxFS) {
    maxFrameSize = maxFS;
    minFrameSize = 64 * 8; // bits
    burst = (maxFS + overhead) * maxCmiFrames;
    dataRate = burst / cmiDuration;
  }
}
```

Token Bucket TSpec
- Max Frame Size
- Min Frame Size
- Burst Size
- Data Rate
- Effective Data Rate
- Effective Burst

MSRP TSpec
- CMI Duration
- Max CMI Frames
- Max Frame Size

Summary

- **802.1Qdd:** Allocation of resources based on traffic volume
- Latency is a resource!
  Traffic Volume → Worst-Case Latency Math → Resource reservation
- MSRP TSpec alone is inefficient for distributed & dynamic reservation
  - Overallocation due to missing flexibility and missing fields
  - Not suited for all types of transmission behavior and shapers
  - No fields intended for per-hop TSpec adjustments / corrections (→ clock drifts, …)
- Suggestion: support multiple different TSpec types
  - Communicated by RA Class Templates
  - No switch/case blocks necessary in internal reservation logic
- Next Steps
  - What information is exactly contained in which TSpec?
  - How exactly would the switches use/adjust that information?
  - Signaling and concrete state machines for the switch logic?

Thank You!

Icons: Rudez Studio (https://www.iconfinder.com/Ruslancorel), Shawn Rubel (https://www.iconfinder.com/Vecteezy), own creations