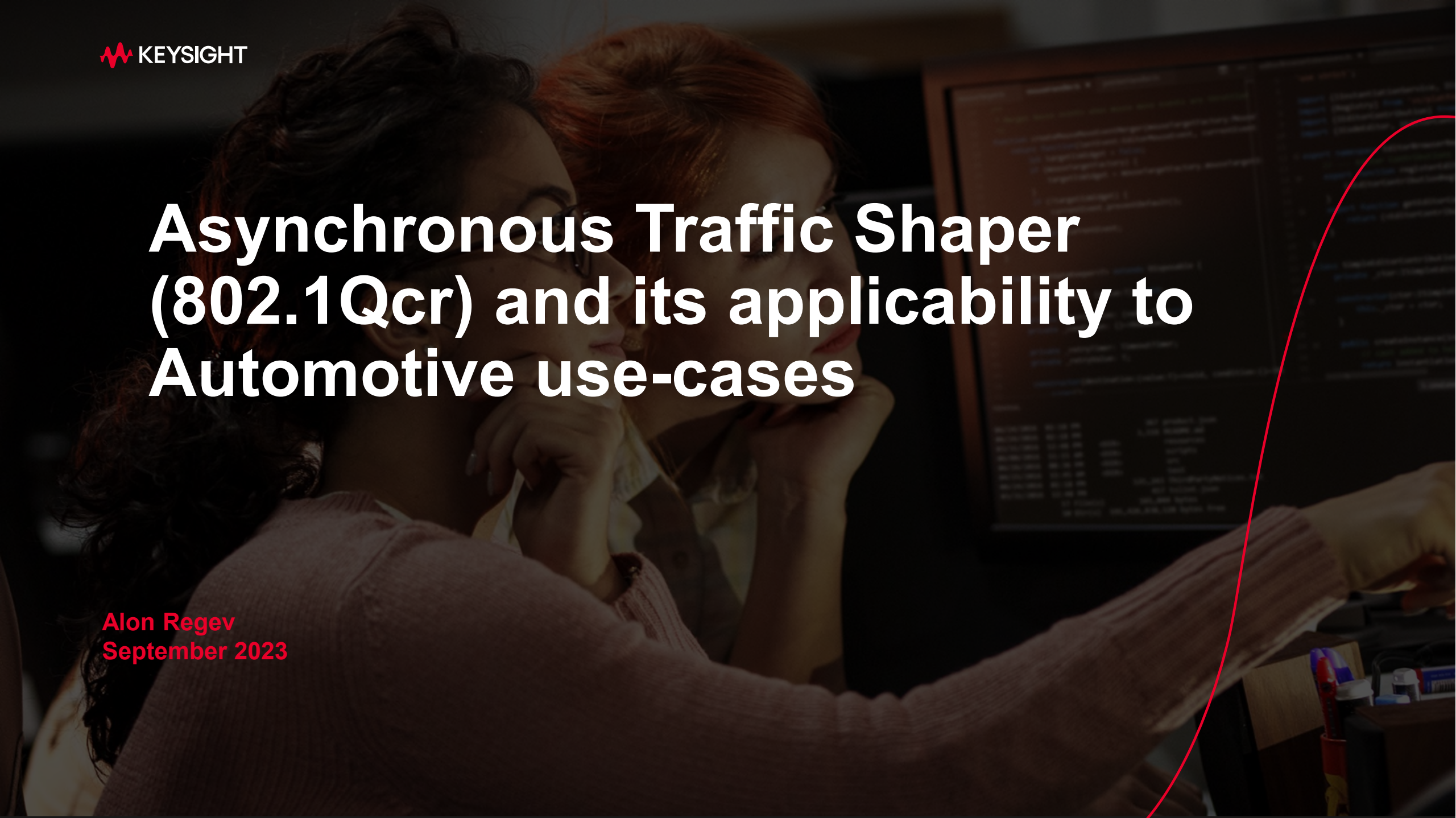


Asynchronous Traffic Shaper (802.1Qcr) and its applicability to Automotive use-cases

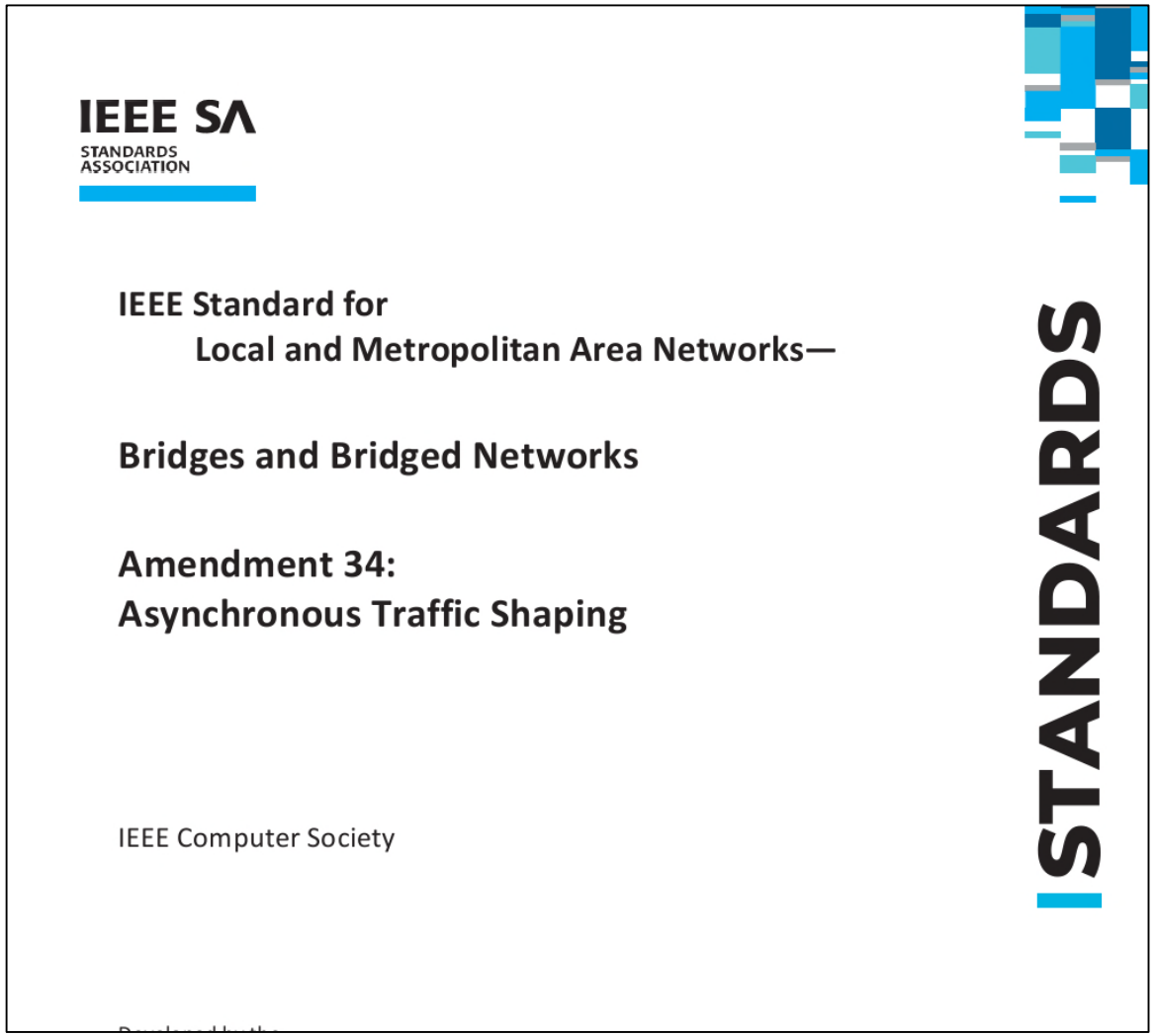
Alon Regev
September 2023

Agenda

- 802.1Qcr ATS (Asynchronous Traffic Shaper) introduction
- Differences between talkers and switches
- Comparison between ATS vs. other TSN schedulers
- Interactions of ATS with other shapers
- Applicability to automotive applications
- Testing ATS for conformance and performance
- Summary and recommendations

Asynchronous Traffic Shaper - Introduction

- Defined in IEEE 802.1Qcr-2020
- Enables maximum latency guarantees
- Does not require time synchronization
- Provides per-stream egress metering of traffic
- Provides fault isolation
- Nodes and/or streams can be dynamically added, changed, or removed at runtime
- Efficient use of Ethernet bandwidth



The scheduler algorithm

For a single stream…

- Based on the Token Bucket Algorithm with some tweaks to minimize hardware complexity
- Applied on each egress port
- Pseudo-code on right is generalized model*
  - 802.1Q defines this procedure in clause 8.6.11.3 for bridges and 49.1.2 for endstations

```
def ProcessFrame(frame):
  lengthRecoveryDuration = length(frame)/ CommittedInformationRate
  emptyToFullDuration = CommittedBurstSize / CommittedInformationRate
  schedulerEligibilityTime = BucketEmptyTime + lengthRecoveryDuration
  bucketFullTime = BucketEmptyTime + emptyToFullDuration
  eligibilityTime = max(arrivalTime(frame), schedulerEligibilityTime)
  if not bridge or (eligibilityTime <= (arrivalTime(frame) + MaxResidenceTimeInS)):
    # The frame is valid
    if (eligibilityTime < bucketFullTime) ?
      BucketEmptyTime = schedulerEligibilityTime :
      BucketEmptyTime = schedulerEligibilityTime + eligibilityTime – bucketFullTime
    Send(frame) at time(eligibilityTime)
  Else:
    # The frame is invalid
    Discard(frame)
```

Details:

Parameters (set during configuration):

- CommittedInformationRate: the “bandwidth” allocated to the steam
- CommittedBurstSize : the maximum amount of data to hold in the bucket
  - Data egress can exceed the CommittedInformationRate only for CommittedBurstSize amount of data
- MaxResidenceTime: (for bridges) the maximum time between ingress and egress of a frame
  - (in bridge) If the eligibilityTime for a frame would exceed the MaxResidenceTime, the frame is dropped

Variables (maintain the state while running):

- eligibilityTime: the earliest time the frame is allowed to egress
- BucketEmptyTime: the “state” of the scheduler; indicates a point in time when the bucket would be empty if the packet were transmitted at the eligibilityTime. May be in the past, present, or future.

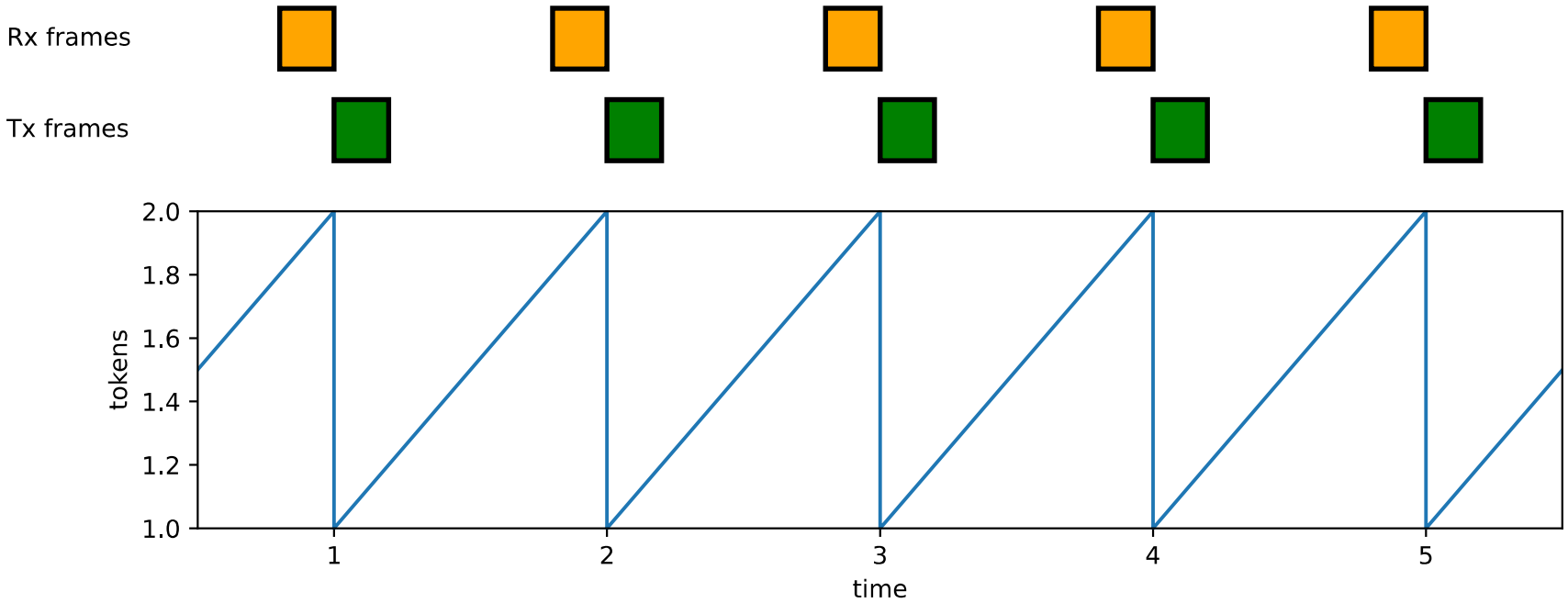
* Scheduler groups are not modeled in this single stream code for simplicity; see following slides for discussion of scheduler groups

The algorithm: example

for a single stream, when the stream ingress rate equals the CommittedInformationRate

- Ingress rate = 200 bits / s (frame length = 200 bits, frame rate = 1 / sec)
- Line rate = 1000 bits/s
- CommittedInformationRate = 200 bits/s
- CommittedBurstSize = 400 bits
- MaxResidenceTimeinS = 10

- “tokens” is the time delta between arrivalTime and BucketEmptyList (to show tokens available)

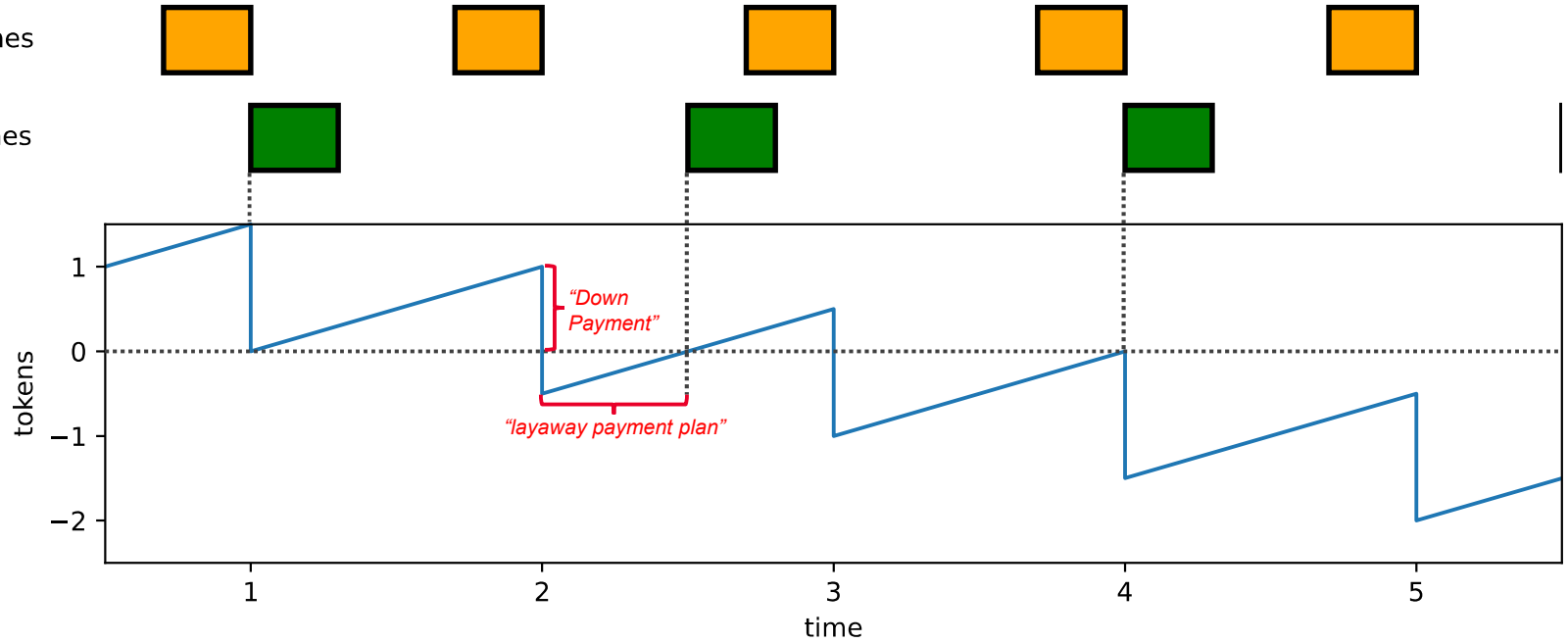


The algorithm: example

for a single stream, when the stream ingress rate is higher than the CommittedInformationRate

- Ingress rate = 300 bits / s (frame length = 300 bits, frame rate = 1 / sec)
- Line rate = 1000 bits/s
- CommittedInformationRate = 200 bits/s
- CommittedBurstSize = 400 bits
- MaxResidenceTimeinS = 10

- “tokens” is the time delta between arrivalTime and BucketEmptyList (to show tokens available)

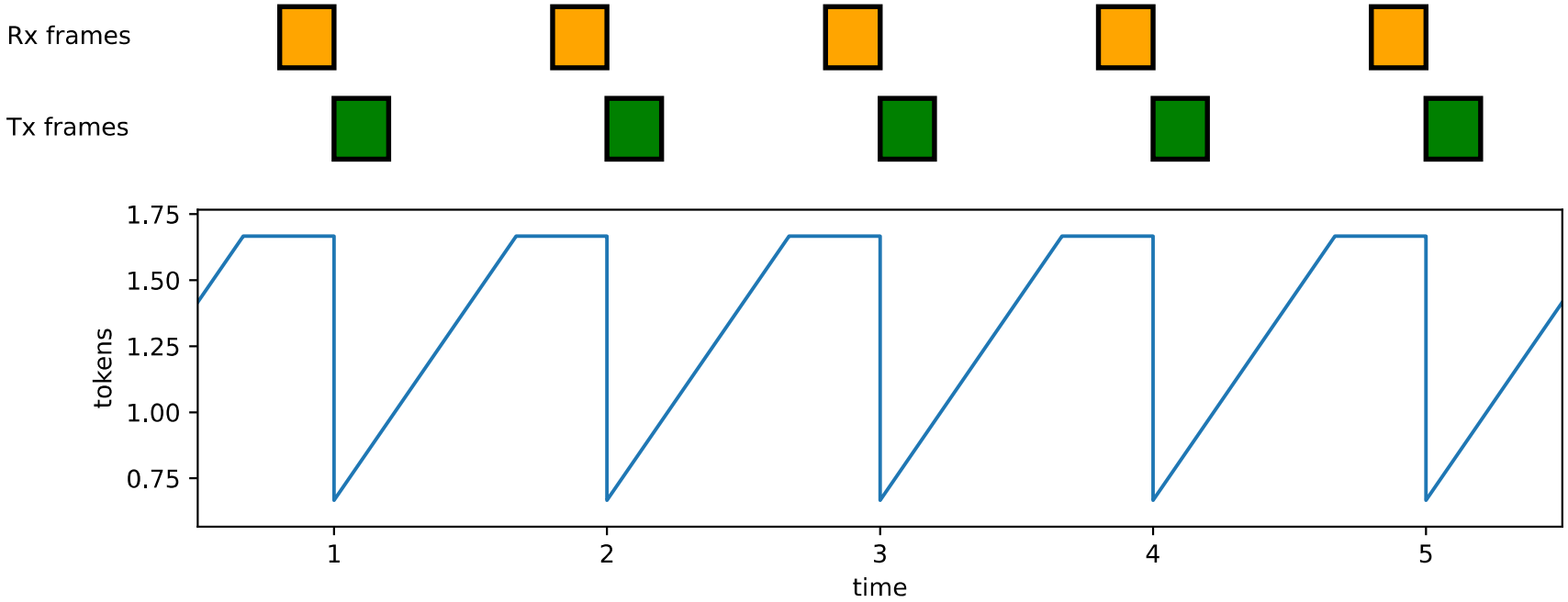


The algorithm: example

for a single stream, when the stream ingress rate is lower than the CommittedInformationRate

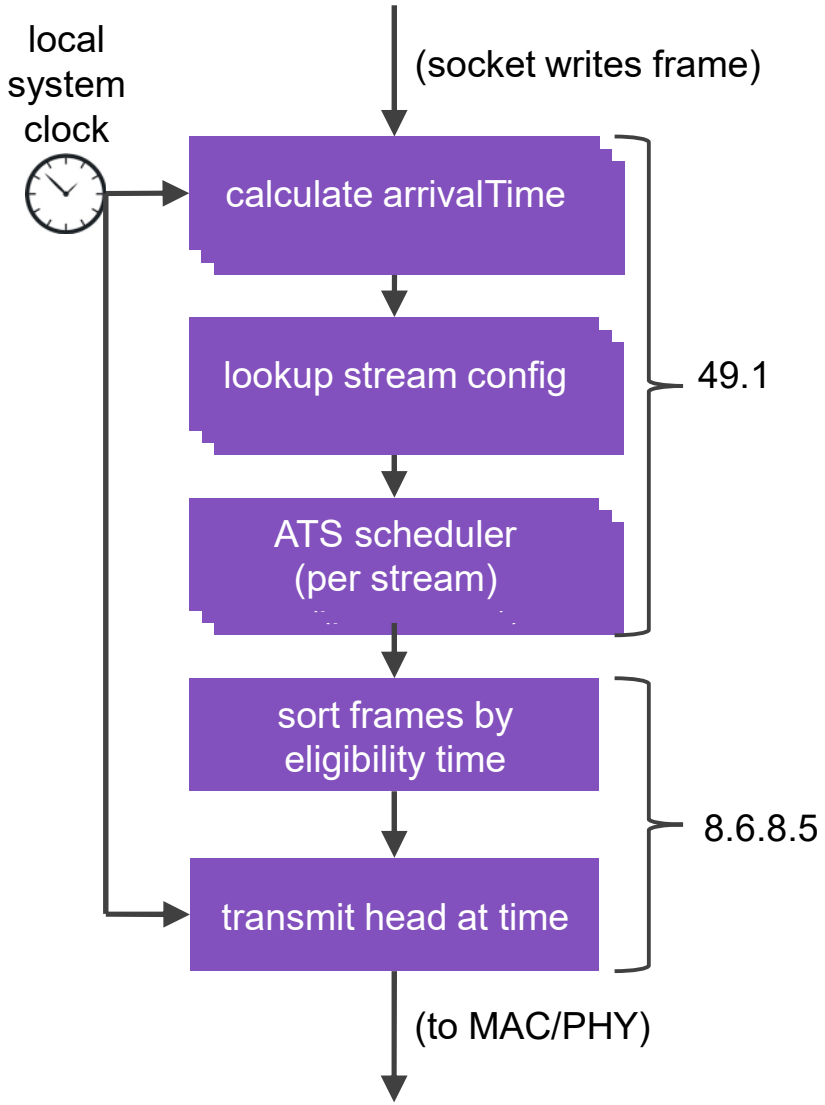
- Ingress rate = 200 bits / s (frame length = 200 bits, frame rate = 1 / sec)
- Line rate = 1000 bits/s
- CommittedInformationRate = 300 bits/s
- CommittedBurstSize = 400 bits
- MaxResidenceTimeinS = 10

- “tokens” is the time delta between arrivalTime and BucketEmptyList (to show tokens available)



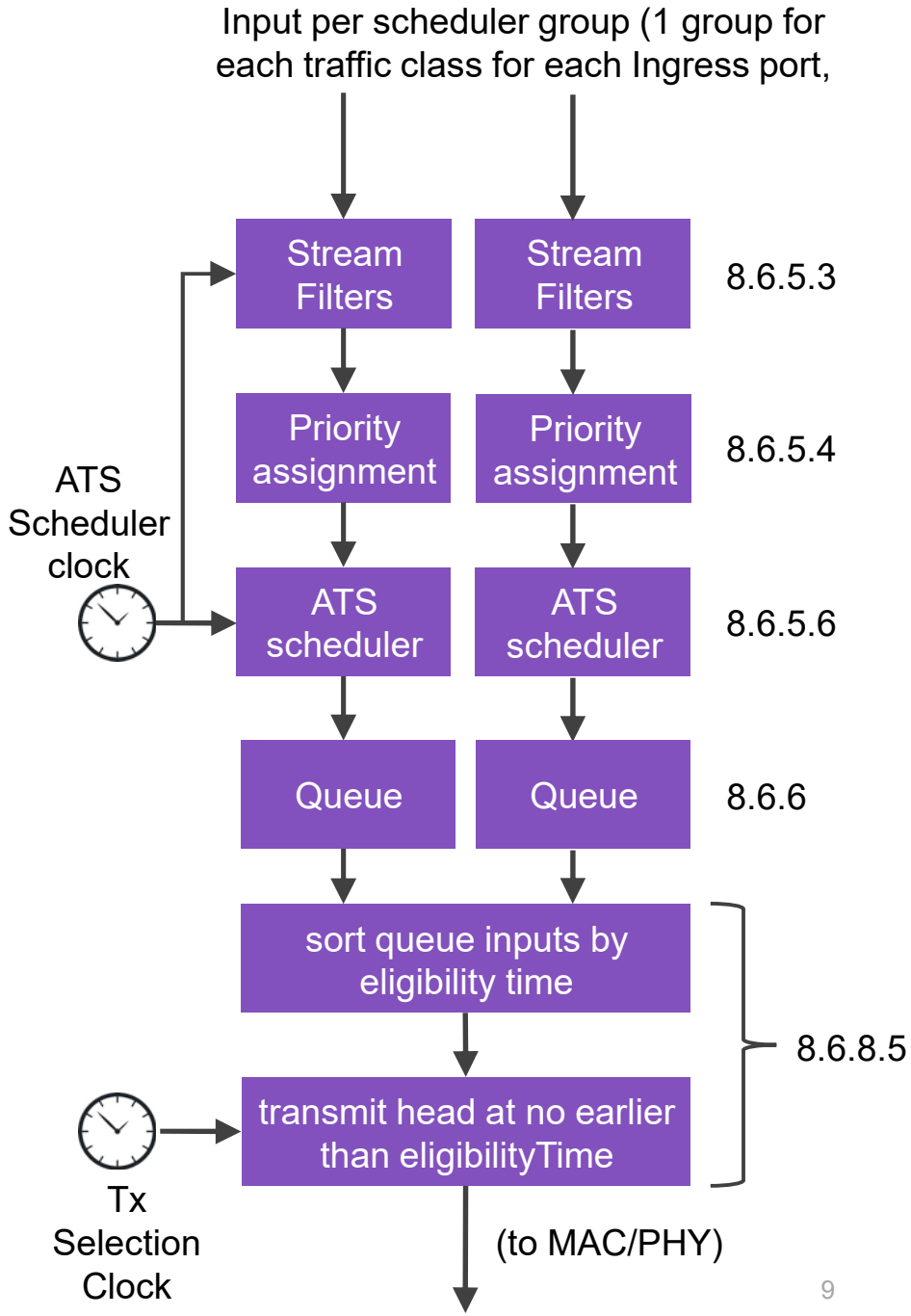
ATS in end stations

- End stations have a simplified ATS implementation defines in 802.1Q clause 49
- A single local clock is used
- One ATS scheduler is assigned per stream
- Frames output by multiple ATS schedulers need to be sorted by eligibility time to determine transmit order
- Frames are never dropped due to the ATS algorithm in end stations
  - In reality there are likely limits, but not specified in 802.1Q
- Can be implemented fully in software (no special NIC functionality is required)



ATS in bridges

- Defined in 802.1Q clause 8.6
- One ATS scheduler instance per scheduler group
  - A scheduler group is shared by all streams from a single ingress port, optionally per stream class
  - Reduces the sorting complexity for ATS outputs (from an ATS scheduler per stream to an ATS scheduler per group)
- Supports priority reassignment based on stream
  - Allows the same stream to be given different priorities in different bridges providing more fine-grained control
- MaximumSDU filtering drops frames greater than per-stream limit
- ATS scheduler drops frames that would not meet MaximumResidenceTime
  - MaximumResidenceTime is configured per scheduler group
- Allows separate scheduler and transmission selection clocks with device-specified maximum offset and frequency deviations



Implications of Scheduler Groups

- In bridges, a scheduler group is shared by all streams from a single ingress port, optionally per stream class
- Frames are not reordered within a stream group
  - if a frame from stream A is received and has an eligibilityTime assigned, subsequent received frames from all other streams will have elibilityTimes set no earlier than this
- A MaximumResidenceTime value is shared by all streams in the stream group
  - Effectively limits the maximum among of time that the eligibilityTime of one stream can delay frames from other streams in the same scheduler group
- Example:
  - Stream A is configured for 1 frame/s; stream B is configured for 100 frames/sec
  - MaximumResidenceTime is set to 100ms
  - What happens if a frame from Stream A arrives early?
    - If the frame arrives more than 100ms before it's schedulerEligibilityTime it is dropped
    - If the frame arrives within 100ms from it's schedulerEligibilityTime, it is queued
      - Frames from stream B will be queues behind it (until the stream A frame's eligibilityTime is reached; up to 100ms)
      - Frames queued for from stream B for the up to 100ms will then be burst after the frame from Stream A

Comparing ATS to other Schedulers

	802.1Qcr Asynchronous Traffic Shaper	802.1Qav Credit Based Shaper	802.1Qbv Ehnancements for Scheduled Traffic
Requires 802.1AS time sync	No, but devices need to specify maximum frequency deviation (i.e., 100ppm)	Yes, but only frequency syntonization is really required	Yes - gate time accuracy depends on time sync accuracy
Shaping unit	Shaping per stream; scheduler groups affect shaping	Shaping per stream (at talker) and per class	Shaping per gate
Burst Accumulation	only within stream group	yes	Bursts are allowed only when gate open, protecting other traffic classes
Maximum latency guarantees	Yes	No - failure cases have been demonstrated	Yes
Traffic types	Periodic, periodic burst, random with bandwidth limit	Only fixed periodic traffic	Periodic, periodic burst, random with bandwidth limit
Relative Max Latency	Medium	High	Low
Relative complexity of Implementation	Medium	Low	High
Relative complexty of configuration	Medium	Low	High

Interactions of ATS with other schedulers

- ATS scheduled based on receive time; not based on transmit status
  - If ATS traffic egress is blocked (by higher priority queues, gates, etc.) transmit may be delayed (while scheduling is not)
- Interactions with 802.1Qbv Enhancements for Scheduled Traffic
  - ATS will keep on scheduling frames (based on eligibilityTime) while the gate associated with the ATS queue is closed
  - When the gate is opened, ATS traffic can send a burst
- Interactions with 80.1Qav Credit Based Shaper
  - CBS is intended to use the highest priority queues (but will only transmit if credits are available)
  - ATS in intended to use the highest N queues (for N priorities)
  - Each can block the other if given higher priority (leading to a potential burst from frames scheduled by the other algorithm

ATS applicability to Automotive

- Two distinct types of systems
  - In-car, engineered networks with (mostly) predetermined latency / bandwidth requirements
  - Changeable control applications such as assembly lines

ATS for in-vehicle networks

- Low-cost to implement
- Guaranteed worst case latency (better than 802.1Qav)
- Better utilization of available bandwidth than 802.1Qbv
- Better performance for the next-to-highest priority traffic classes
- Does not require time synchronization or hardware timestamping
- Unlike 802.1Qbv that only supports periodic traffic, ATS can be used for periodic, event-based, and rate-constrained traffic

ATS for control applications

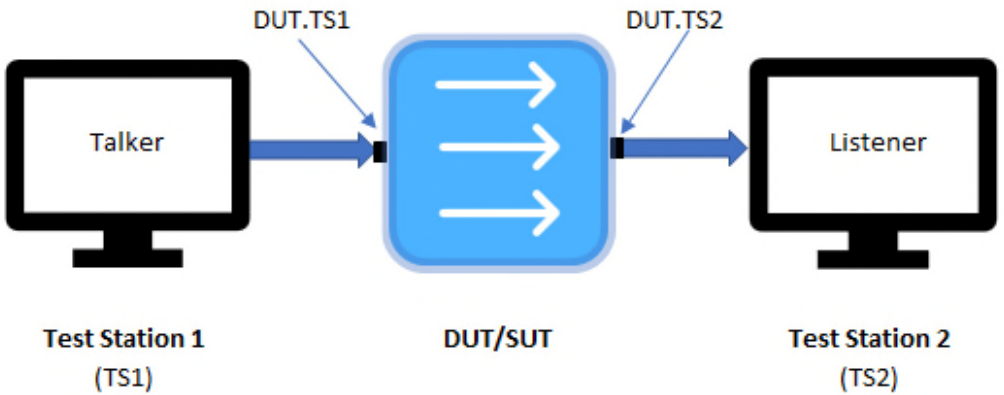
- Low planning / implementation effort
- Easier to modify than 802.1Qbv
  - Only need to add new stream to table
- Better utilization of available bandwidth than 802.1Qbv
- Supports many types of traffic / streams with different constraints on each stream
- Does not require time synchronization or hardware timestamping
- Better performance for the next-to-highest priority traffic classes

What needs to be tested in ATS

- Device-level conformance Testing for ATS should include
  - Stream Filtering: correctly identifying the Stream Gate, scheduler ID, and Max SDU; maximum SDU filtering
  - Stream Gating: correctly updating (or not updating) the priority
  - Proper handling of multiple priorities
  - ATS scheduler: eligibilityTime assignment, MaxResidenceTime enforcement, GroupEligibilityTime enforcement
  - Clock offsets are within the ClockOffsetMin to ClockOffsetMax range
  - Clock rates are within the ClockRateDeviationMax
  - assignedEligibilityTime: property compensating for clock offset and processing delay
  - Dynamic changes to configuration parameters
- System and Interoperability tests should be run to validate interactions between ECUs

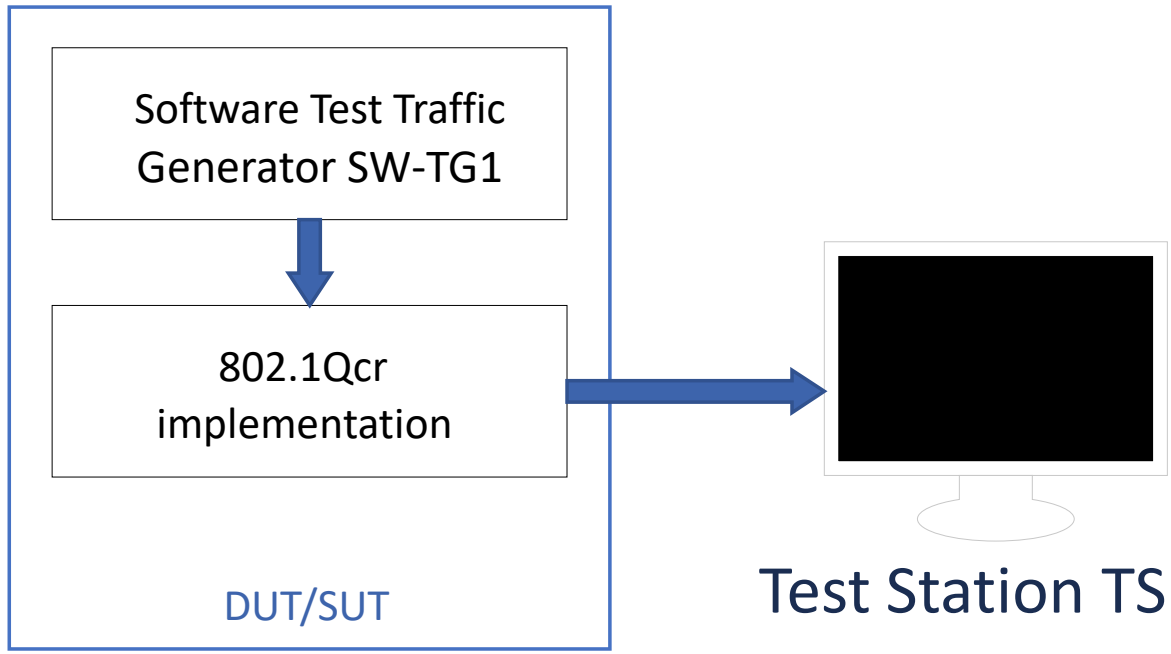
Validating an ATS bridge

- To fully test 802.1Qcr, both the traffic to the 802.1Qcr component and the traffic transmitted from the 802.1Qcr component need to be controlled and monitored by external test station
  - To test multiple scheduler groups, multiple talker ports need to be connected to the DUT
- Traffic needs to be sent to the 802.1Qcr implementation with controlled times of arrival at the 802.1Qcr implementation.
- To validate frequency deviation, the test station must use an accurate reference time source
- All test stations/test ports must be time synchronized



Validating an ATS end station implementation

- To fully test 802.1Qcr, both the traffic to the 802.1Qcr component and the traffic transmitted from the 802.1Qcr component need to be controlled and monitored.
- Traffic from the 802.1Qcr engine is typically analyzed by an external test station
- Traffic needs to be sent to the 802.1Qcr implementation with controlled times of arrival at the 802.1Qcr implementation.
  - A standardized software traffic generator that can be externally controlled is proposed to accomplish this
- To validate frequency deviation, the test station must use an accurate reference time source



Summary and Recommendations

- While 802.1Qcr has been published in 2020, commercial implementations are only starting to become available
- There is limited ATS knowledge / experience in automotive
  - This presentation attempts to take a step is resolving this
- ATS has potential benefits and limitations vs. other schedulers
- It is recommended that
  - ATS be evaluated for applicability to any automotive application
  - System simulations should be used to model traffic behavior under both ideal and unexpected conditions (i.e., blabbing idiot or out-of-spec oscillator)
  - Any ATS implementation should be tested to validate it's conformance, performance and interoperability

KEYSIGHT

Questions?

Alon Regev
alon.regev@keysight.com