Always On, Every Packet
CSIG: Congestion Signaling in the AI Era

Jai Kumar

AIDC Side Meeting @IETF124
06 Nov 2025

BROADCOM®

Topics

- Motivation
- CSIG Domain and Logical Functions
- CSIG Tag Format and Layering
- CSIG Signals and Quantization
- Usecases and UET simulation

Workloads: Era of Extreme Network Demands

- Continuing trends in the AI era: Horizontal scaling is inevitable
  - Extreme *reliability, performance and efficiency* requirements for scale-up and scale-out networks serving AI workloads
  - AI workloads are extremely bandwidth-hungry and tail latency-intolerant
- New norms for network congestion in AI workloads

Massive, synchronized bursts that amplify as the network fabric scales	Congestion events that manifest at sub-millisecond timescales on network switches	Predictable and repeating patterns of short-lived congestion

- Many control loops operate at different timescales to
  - Efficiently utilize available network capacity *at fine-grained timescales*
  - Enable tight guarantees on tail latency and throughput for collectives

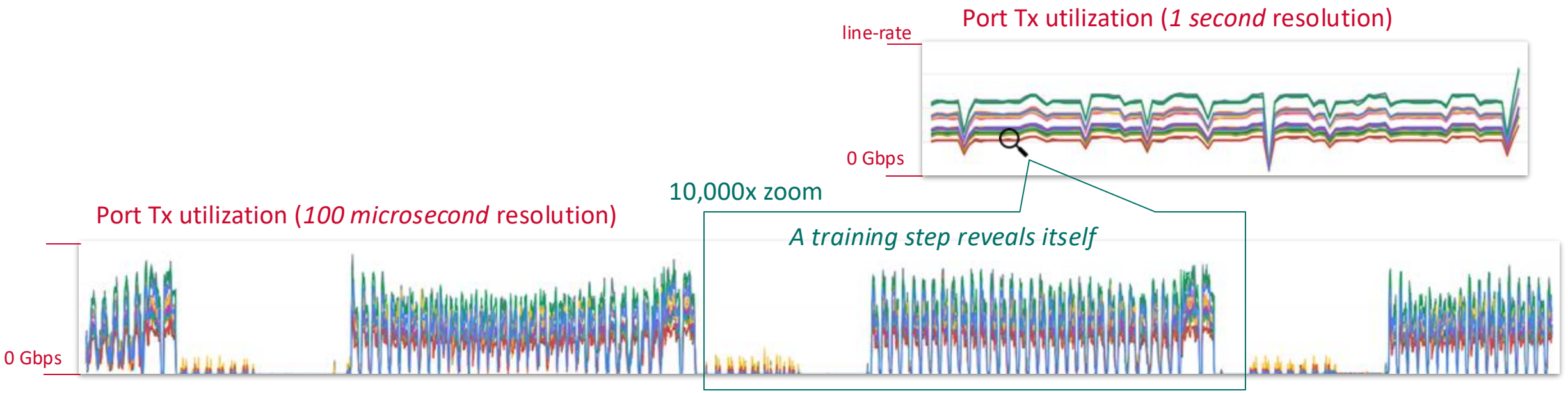
Congestion control, load balancing, multipathing, scheduling, traffic engineering, provisioning

Central observation: *Accurate and fine-grained congestion signals* needed for observability and control

BROADCOM

High-resolution network signals are *necessary*

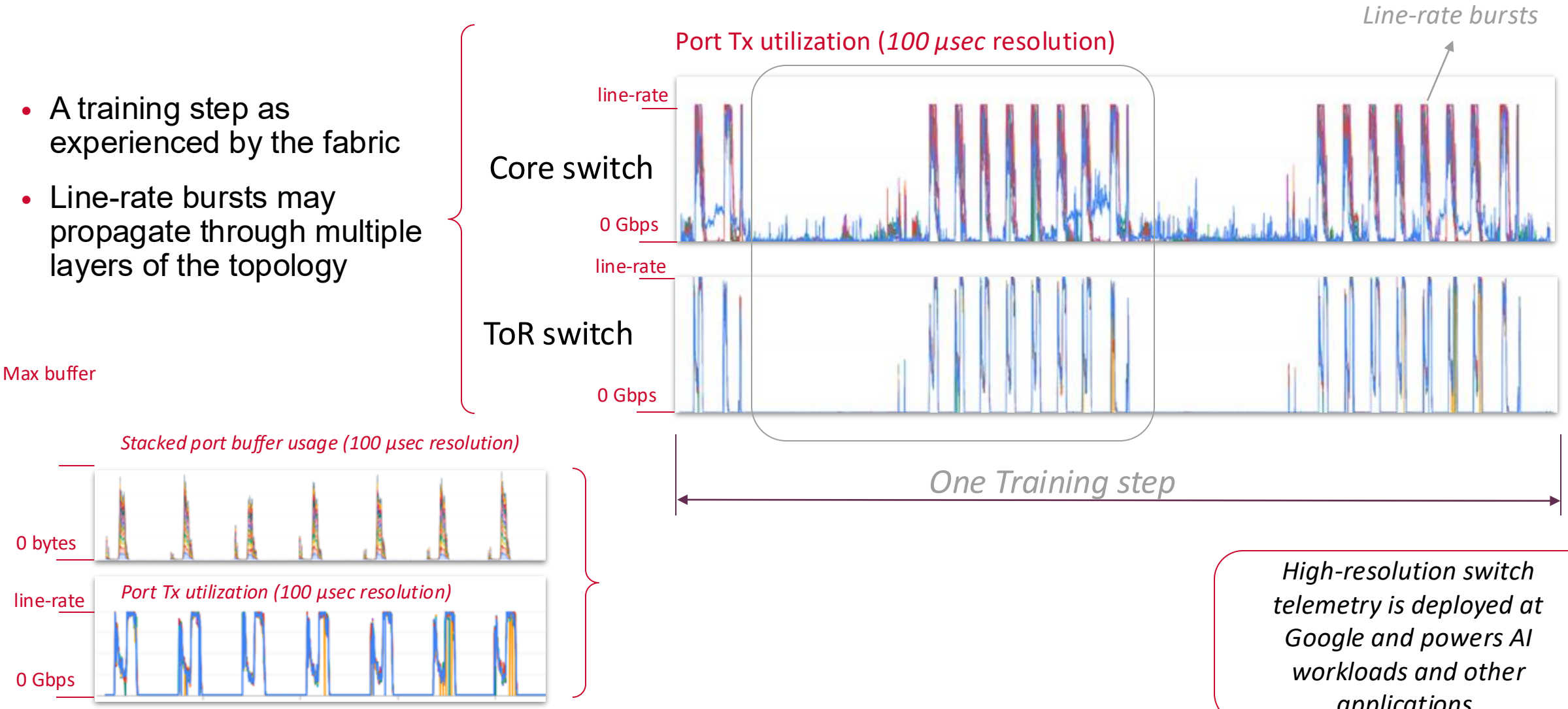
- Accurately detecting congestion *locally* on a switch requires signal measurements at sub-millisecond timescales
- Real-world example from a GPU ToR at Google:
  - Shifting from 1-second to 100-µsec telemetry exposes the fine-grained, repeating congestion patterns and idle gaps inherent to AI workloads



BROADCOM

Network Signature of a Training step

- A training step as experienced by the fabric
- Line-rate bursts may propagate through multiple layers of the topology



High-resolution switch telemetry is deployed at Google and powers AI workloads and other applications.

BROADCOM

CSIG - Simple & Efficient In-band Signaling Protocol

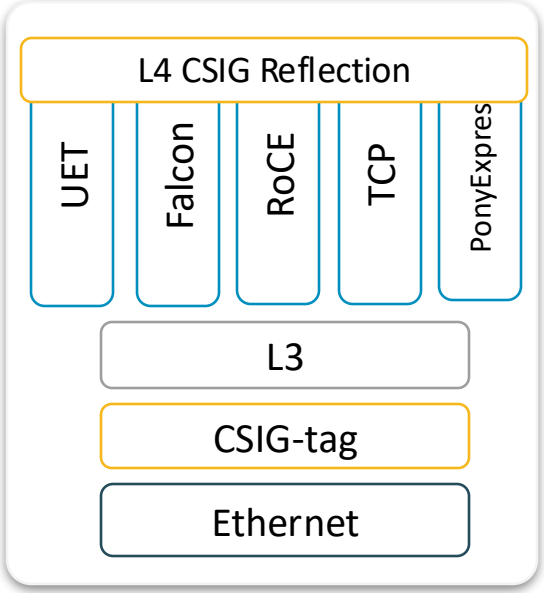
Packet format (with reflection)

Dest MAC | Src MAC | **CSIG tag** | EtherType | IP header | Transport | **CSIG Reflection** | Payload

CSIG-tag Value Structure

Signal Type	Signal Value	Locator Metadata
Set / requested by end host. Oneof: • min (ABW/C) • min (ABW) • max (PD) • max (nQD) …	Bucketized or Quantized signal value	Indicator of the bottleneck location

Signals are computed at high-res timescales on switches



- ✓ Per-hop signal scope pinpoints congestion state at bottleneck
- ✓ Space-efficient telemetry collection
- ✓ Compute-efficient telemetry collection
- ✓ In-band collection interpretable in application/transport context
- ✓ Transport- and encapsulation-agnostic
- ✓ Deployable on pre-CSIG switch silicon
- ✓ Extensible

BROADCOM

CSIG Domain

In the CSIG Domain, the UE fabric can interpret and update CSIG tags in accordance with pre-defined logical functions

The following logical functions comprise the CSIG Domain

- CSIG Init
- CSIG Transit
- CSIG Term and Reflect
- CSIG Strip and Forward

CSIG Init

Init is always done by the NIC, aka UE FEP

- FEP inserts CSIG tag per policy and forwards the packet, either:
  - in every packet (signals can be alternated)
  - in 1/N packets, or
  - in CSIG-created packets, also referred to as probes
- CSIG tag signal value ‘S’ is initialized appropriately
  - Specification will define the init signal value ‘S’ for each signal type ‘T’
  - e.g., maxDelay init value is ‘0’, minBW is ‘0xff’

CSIG Transit

UE Fabric switch performs a transit function per policy

- ‘Compare and Update’

or

- No ‘Compare and Update’ (e.g. TRIM packets)

or

- ‘Strip and Forward’

CSIG Update sequence

– Compare and Update the CSIG Tag (CSIG::Signal value, CSIG::Locator Metadata)

– CSIG::T == ‘min-ABW[/C]’
  If LOCAL_ABW[/C] < CSIG::SIGNAL
    CSIG::SIGNAL = LOCAL_ABW[/C]
    CSIG::LM = LOCAL_LM

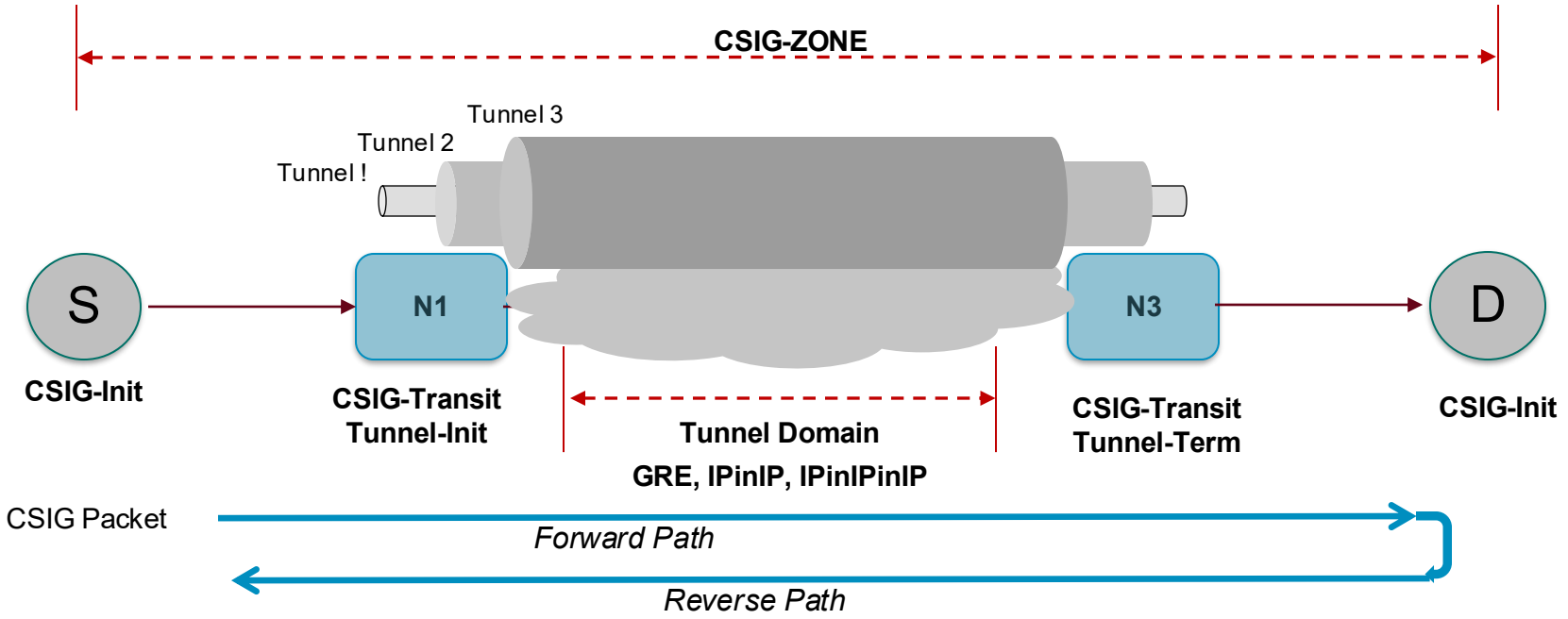
– CSIG::T == ‘max-Delay’
  If LOCAL_DELAY > CSIG::SIGNAL
    CSIG::SIGNAL = LOCAL_DELAY
    CSIG::LM = LOCAL_LM

CSIG Term and Reflect

UE NIC/FEP terminates the CSIG Domain

- Term nodes do not update the CSIG tag signal value ‘S’
- Term only: CSIG tag removed and consumed by application layer
  - Applicable to receiver-based congestion control algorithms
- Term and Reflect: CSIG tag removed and reflected to CSIG Init node
  - Application to send-based congestion control algorithms

Congestion Control Topology and Packet Flow

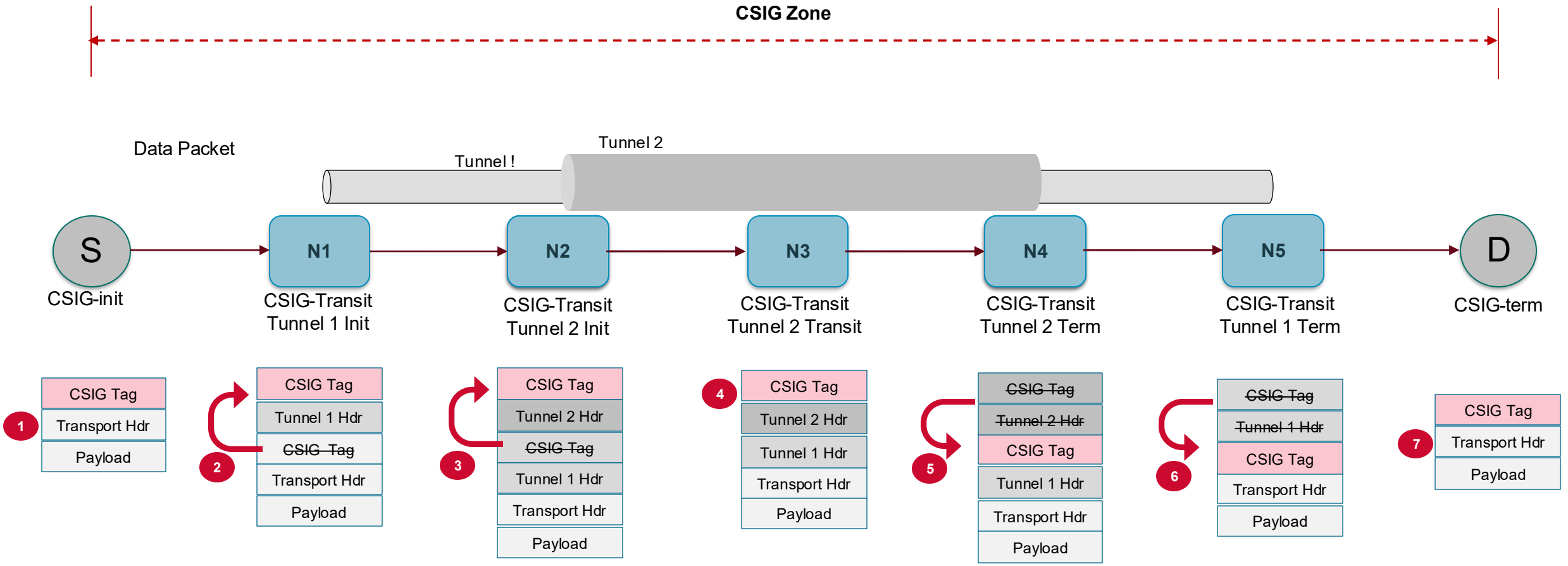


- *CSIG-Init: NIC sending CSIG packet (insert CSIG tag with init values)*
- *CSIG-Transit: Network Device updating the CSIG tag fields in CSIG packet*
- *CSIG-Term: NIC consuming the CSIG packet; will reflect ACK packet with CSIG data from incoming packet*

NIC is always the CSIG init and term

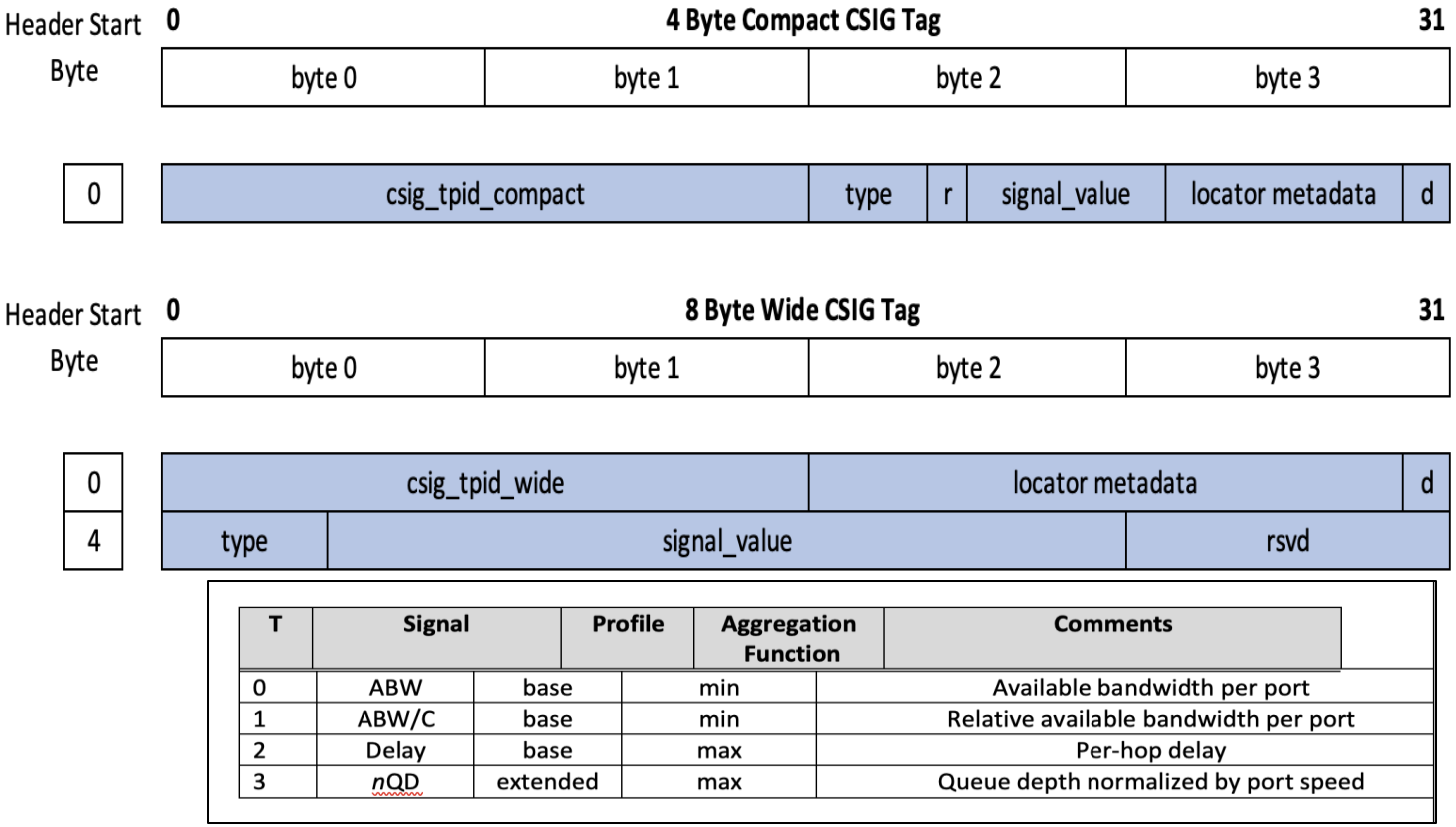
Network device MAY be a CSIG term(strip-and-forward) when terminating a CSIG zone

Tunnel in Tunnel CSIG Tag Placement Requirement



1. *NIC Inserts CSIG Tag*
2. *N1 updates and copy CSIG Tag to tunnel 1*
   N1 deletes incoming CSIG Tag from transport header
3. *N2 updates and copy CSIG Tag to tunnel 2*
   N1 deletes incoming CSIG Tag from transport header
4. *N3 updates CSIG Tag;*
5. *N4 performs tunnel 2 decap and copies updated CSIG Tag in tunnel 1 header*
6. *N5 performs tunnel 1 decap and copies updated CSIG Tag to non tunneled packet transport header*
7. *NIC consumes CSIG Header*

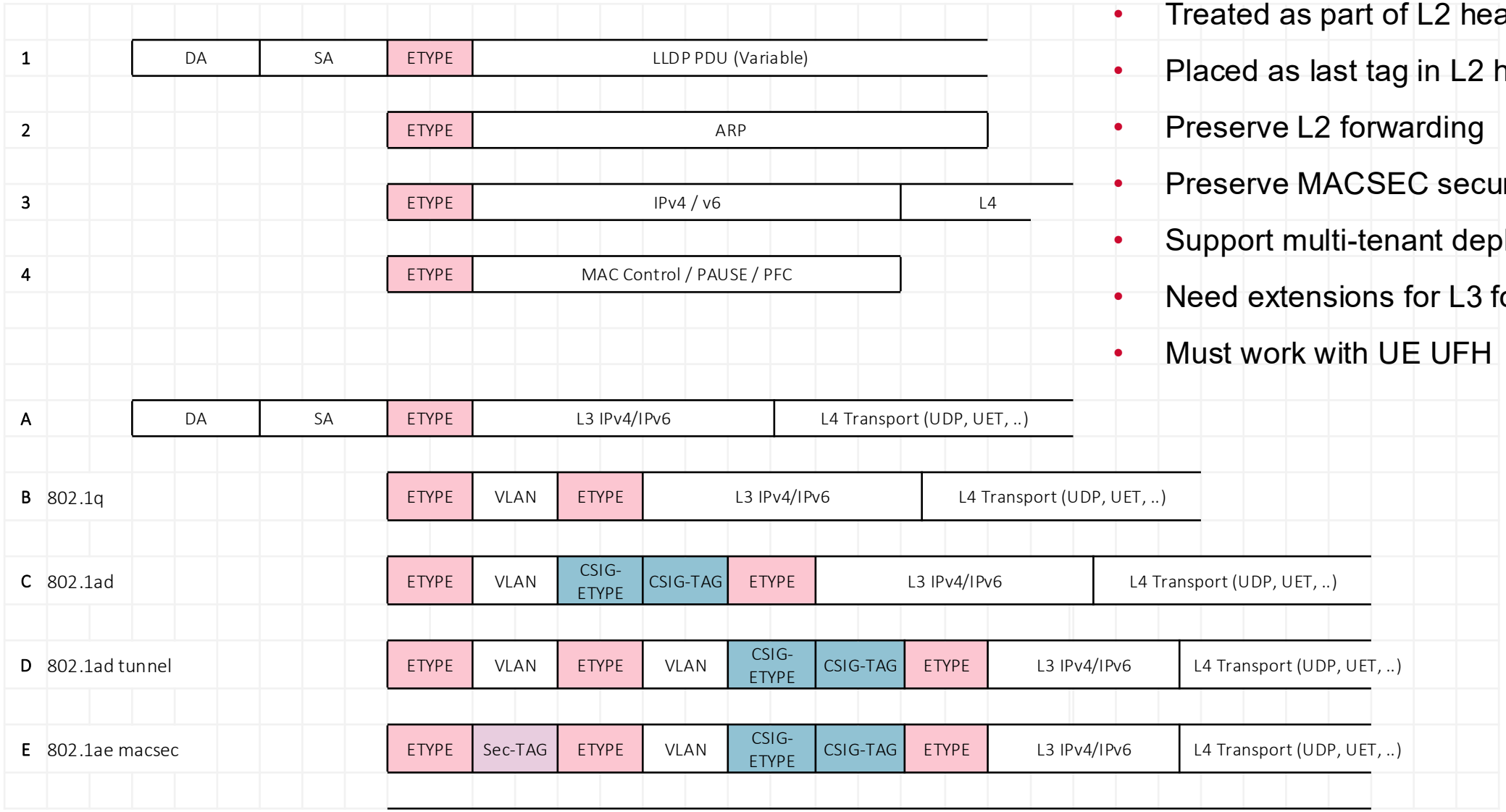
Wire Format



T	Signal	Profile	Aggregation Function	Comments
0	ABW	base	min	Available bandwidth per port
1	ABW/C	base	min	Relative available bandwidth per port
2	Delay	base	max	Per-hop delay
3	*n*QD	extended	max	Queue depth normalized by port speed

csig_tpid_compact/wide: This is the IEEE allocated ether type
type (T): This is the signal type indicating what kind of telemetry data to collect
signal_value (S): This is the quantized value of the signal after the operation of compare and update
locator_metadata: This field is updated by the hop that updates the signal value based on compare and update function
trim bit (D): This bit is set if the packet undergoes TRIM

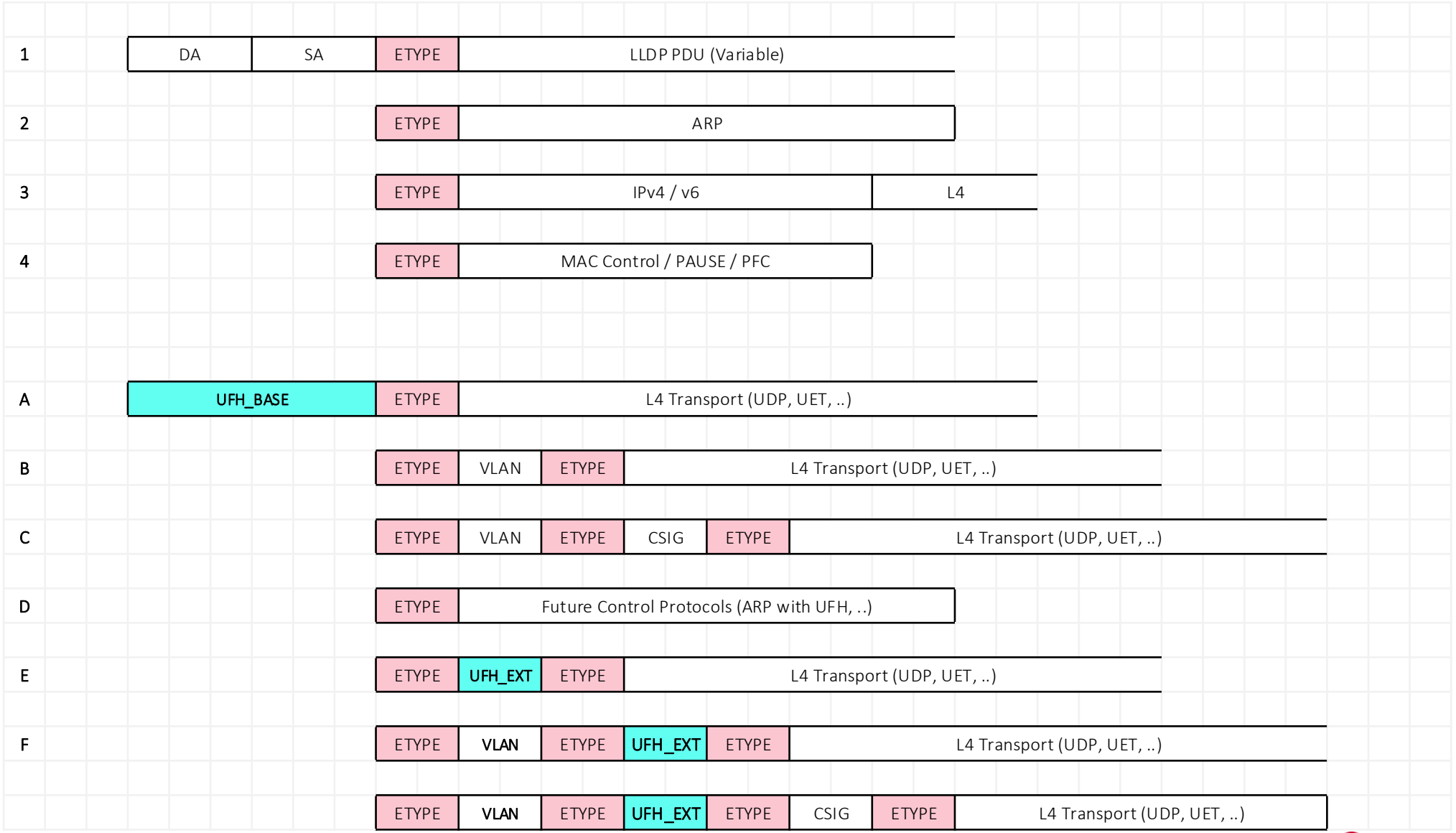
CSIG tag stacking



- Treated as part of L2 header
- Placed as last tag in L2 header
- Preserve L2 forwarding
- Preserve MACSEC security
- Support multi-tenant deployment
- Need extensions for L3 forwarding
- Must work with UE UFH

BROADCOM

CSIG tag stacking with UFH



BROADCOM

Raw ABW Algorithm

$m = Traffic\ in\ Bytes\ measured\ over\ a\ time\ interval\ t$
$p = port\ speed(Bps)$

$$rate\ r = \frac{m}{t}(Bps)$$
$$Available\ BW = (p - r)$$
$$Quantized\ Available\ BW :: Q(p - r)$$

$$pkt \rightarrow minBW = \min(Quantized\ Available\ BW, pkt \rightarrow minBW)$$

All network devices in a CSIG Domain must be configured with the same value for t and the same quantization range

m is the tx bits seen on the wire as accounted for the port statistics
Any overhead from Link Layer is not included in ABW algo. Overhead of Link Layer need to be accounted by the end point for CC algorithm
ABW should account for data frames that can be controlled by the sender for a given flow rate

[MAC generated frames e.g. PAUSE. LLR, CBFC frame or PFC frame are not include]

Raw ABW/C Algorithm

$m = Traffic\ in\ Bytes\ measured\ over\ a\ time\ interval\ t$
$p = port\ speed$ in Bps

$$rate\ r = m/t$$
$$Consumed\ Load = ({}^{r}/_{p}) * 100$$
$$Available\ Load = 100(1 - ({}^{r}/_{p}))$$
$$Quantized\ Available\ Load :: Q(Available\ Load)$$

$$pkt \rightarrow minLoad = \min(Quantized\ Available\ Load, pkt \rightarrow minLoad)$$

All network devices in a CSIG Domain must be configured with the same value of t and the same quantization range

m is the tx bits seen on the wire as accounted for the port statistics
Any overhead from Link Layer is not included in ABW algo. Overhead of Link Layer need to be accounted by the end point for CC algorithm
ABW should account for data frames that can be controlled by the sender for a given flow rate

[MAC generated frames e.g. PAUSE. LLR, CBFC frame or PFC frame are not include]

max(Delay): Maximum Per-Hop Delay

Maximum delay in nanoseconds among observed per-switch delays at switch elements traversed by a given packet's path through the fabric

- Associated Math Function 'max'
- Algorithm Used: Observed value in the switch pipeline
- Per-packet information
  - Forwarding pipeline delay is included
  - Link-layer (MAC/PHY) delay is not included

CSIG Domain need not be time-synchronized

max(nQD): Maximum Normalized Queue Depth

Maximum among queue depths as percentage occupancy at each successive queue traversed along a packet's path, each measured at dequeue time

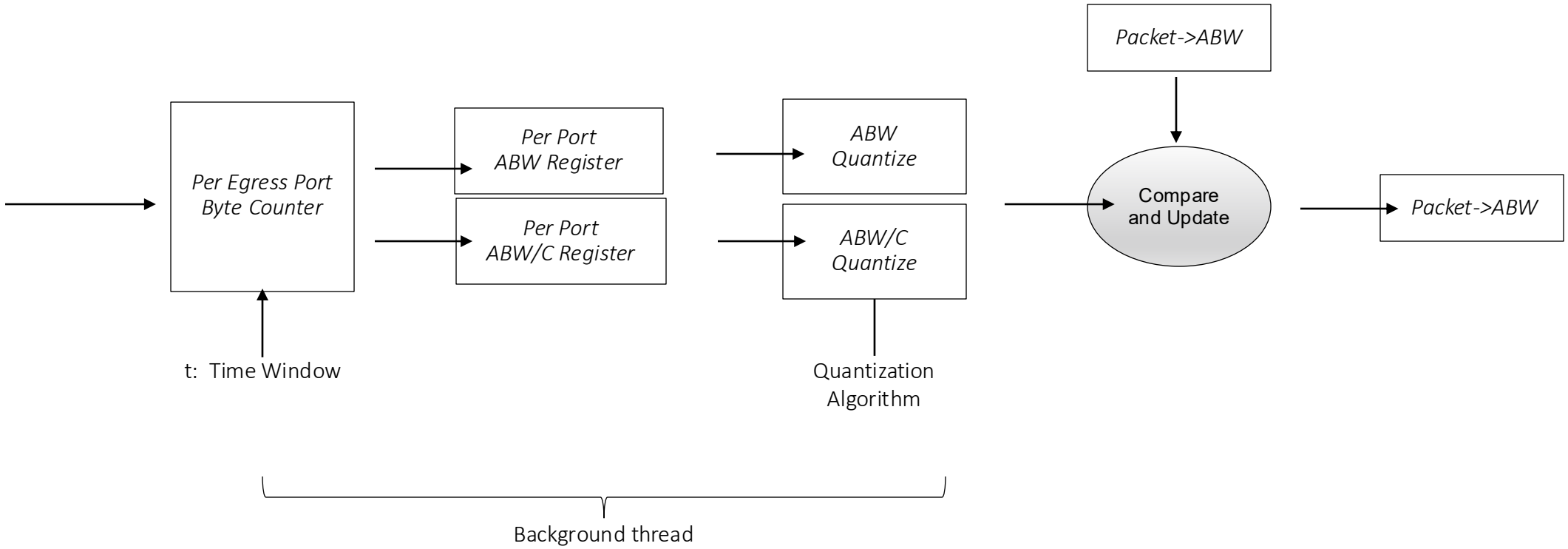
- Associated Math Function 'max'
- Algorithm Used: Observed value in the switch pipeline
- Implementation akin to that for ECN: check queue depth at dequeue time, but record normalized depth value rather than Boolean result of threshold comparison
- Per-packet information
- Leading indicator of congestion (multi bit signal compared to deq-ECN)
- Tolerates non-uniform buffer size across the UE fabric

nQD as a multi-bit signal for a UET multi-path load balancing algorithm that currently uses 1-bit ECN

Enable a sender to implement a weighted load balancing among paths based on the multi-bit nQD values

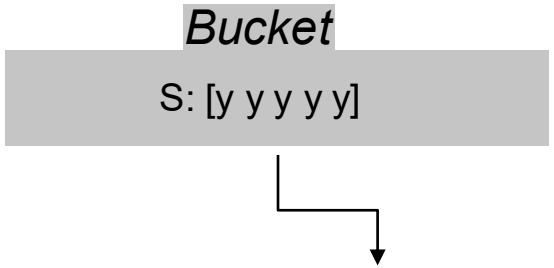
- The UET multi-path load balancing algorithm tries to avoid temporary congested paths by steering packets onto clean paths. Ideally, with the leading congestion signal (carried by nQD) to guide the path selection, the congested paths are skipped appropriately, the Fabric enters a state where the NSCC-window is stable and target queueing delay is well contained, and no packet drop anywhere in the Fabric due to buffer overflow

Signal Quantization Process



BROADCOM

Quantization Algorithm – Compact Header



Bucket	Range
0	[Min, Max]
1	[Min, Max]
2	[Min, Max]
…	[Min, Max]
31	[Min, Max]

- 5 bits provide 32 buckets
- Provide full control over bucket ranges
- 32 range checkers per signal type ‘T’
- Max: 2^3 * 32 range checkers

Quantization – Wide Header

- 20 bits provide 1M buckets
- Prohibitively expensive to allocate 1M range checkers in HW
- Use step function with base value for range allocation to buckets
- Easier to implement and provides fair distribution of ranges over 1M buckets

Quantization Algorithm – Wide Header

Bucket Range Allocation Formula:

$$B_i = bV + (i + 1) * 2^b$$

Where

B_i : i^{th} Bucket

O: Observed Value

bV: base Value

b: Step function

StepFunction and Base Value is always power of 2

Quantization Algorithm – Wide Header

Bucket Range Allocation Example:

StepFunction = 8 or 2^b where b = 3

Bucket	bV=0	bV=8	bV=16
0	<8 units	<16 units	<24 units
1	<16 units	<24 units	<32 units
2	<24 units	<32 units	<40 units
3	<32 units	<40 units	<48 units
4	<40 units	<48 units	<56 units
5	<48 units	<56 units	<64 units
6	<56 units	<64 units	<80 units
7	<64 units	<72units	<96 units

*B[0] = 0 + 1*8 = 8*	*B[0] = 8 + 1*8 = 16*	*B[0] = 16 + 1*8 = 24*
*B[1] = 0 + 2*8 = 16*	*B[1] = 8 + 2*8 = 24*	*B[1] = 16+ 2*8 = 32*
*B[2] = 0 + 3*8 = 24*	*B[2] = 8 + 3*8 = 32*	*B[2] = 16 + 3*8 = 40*
*B[3] = 0 + 4*8 = 32*	*B[3] = 8 + 4*8 = 40*	*B[3] = 16 + 4*8 = 48*

StepFunction = 16 or 2^b where b = 4

Bucket	bV=0	bV=16	bV=32
0	<16 units	<32 units	<48 units
1	<32 units	<48 units	<64 units
2	<48 units	<64 units	<80 units
3	<64 units	<80 units	<96 units
4	<80 units	<96 units	<112 units
5	<96 units	<112 units	<128 units
6	<112 units	<128 units	<144 units
7	<128 units	<144 units	<160 units

*B[0] = 0 + 1*16 = 16*	*B[0] = 16 + 1*16 = 16*	*B[0] = 16 + 1*8 = 24*
*B[1] = 0 + 2*16 = 32*	*B[1] = 16 + 2*16 = 32*	*B[1] = 16+ 2*8 = 32*
*B[2] = 0 + 3*16 = 48*	*B[2] = 16 + 3*16 = 48*	*B[2] = 16 + 3*8 = 40*
*B[3] = 0 + 4*16 = 64*	*B[3] = 16 + 4*16 = 64*	*B[3] = 16 + 4*8 = 48*

Quantization Algorithm – Wide Header

Bucket Range Derivation Formula:

$$B^i = (O - bV) \gg b,$$

Where

B^i : i^{th} Bucket

O: Observed Value

bV: base Value

b: Step function

StepFunction and Base Value is always power of 2

Quantization Algorithm – Wide Header

Bucket Range Derivation Example:

StepFunction = 8 or 2^b where b = 3			
Bucket	bV=0	bV=8	bV=16
0	<8 units	<16 units	<24 units
1	<16 units	<24 units	<32 units
2	<24 units	<32 units	<40 units
3	<32 units	<40 units	<48 units
4	<40 units	<48 units	<56 units
5	<48 units	<56 units	<64 units
6	<56 units	<64 units	<80 units
7	<64 units	<72units	<96 units

bV=0, O=16, 8, 7	*bV=8, O=24, 16, 8*	*bV=16, O=40, 32, 16*
(16-0) >> 3 = B^2	*(24-8) >> 3 = B^2*	*(16-0) >> 3 = B^2*
(8-0) >> 3 = B^1	*(16-8) >> 3 = B^1*	*(8-0) >> 3 = B^1*
(7-0) >> 3 = B^0	*(8-8) >> 3 = B^0*	*(7-0) >> 3 = B^0*

StepFunction = 16 or 2^b where b = 4			
Bucket	bV=0	bV=16	bV=32
0	<16 units	<32 units	<48 units
1	<32 units	<48 units	<64 units
2	<48 units	<64 units	<80 units
3	<64 units	<80 units	<96 units
4	<80 units	<96 units	<112 units
5	<96 units	<112 units	<128 units
6	<112 units	<128 units	<144 units
7	<128 units	<144 units	<160 units

bV=0, O=48,32,8,	*bV=16, O=80, 64, 32*	*bV=32, O=144, 96, 8*
(48-0) >> 4 = B^3	*(80-16) >> 4 = B^4*	*(144-32) >> 4 = B^7*
(32-0) >> 4 = B^2	*(64-16) >> 4 = B^3*	*(96-32) >> 4 = B^4*
(8-0) >> 4 = B^0	*(32-16) >> 4 = B^0*	*(8-32) >> 4 = B^0*

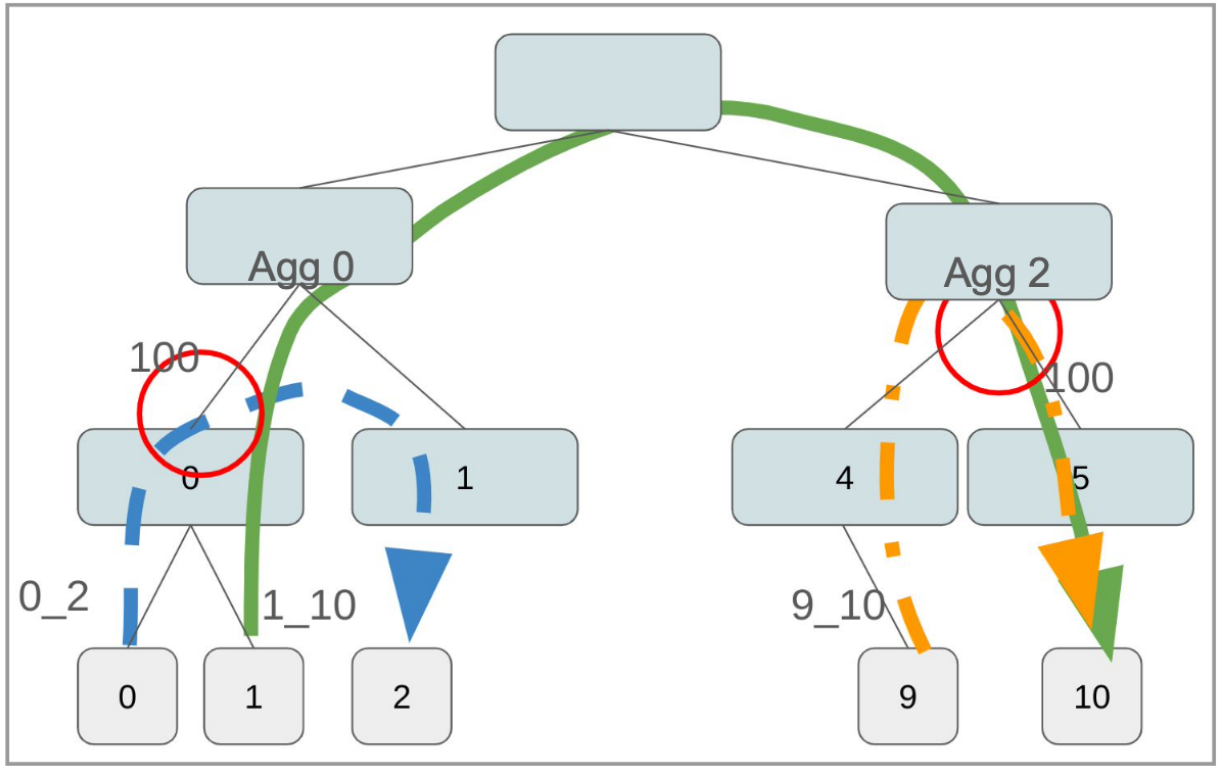
CSIG Signal Usecases and Simulation

CC CSIG Usecases

- min(ABW)
  - Optimal path selection: Choose paths with greater available bandwidth (instead of lumping together all paths that are non-ECN-marked)
- min(ABW/C)
  - Fast adjust of cnwd: Fast ramp up of cwnd within measured available spare capacity, to avoid overshooting (also proposed in HPCC++ Internet Draft)
- max(Delay)
  - Avoid estimated path's baseRTT: Inaccurate measure if queues are already built up
- max(Delay)
  - may better support topologies with less uniform path lengths, such as Dragonfly topologies

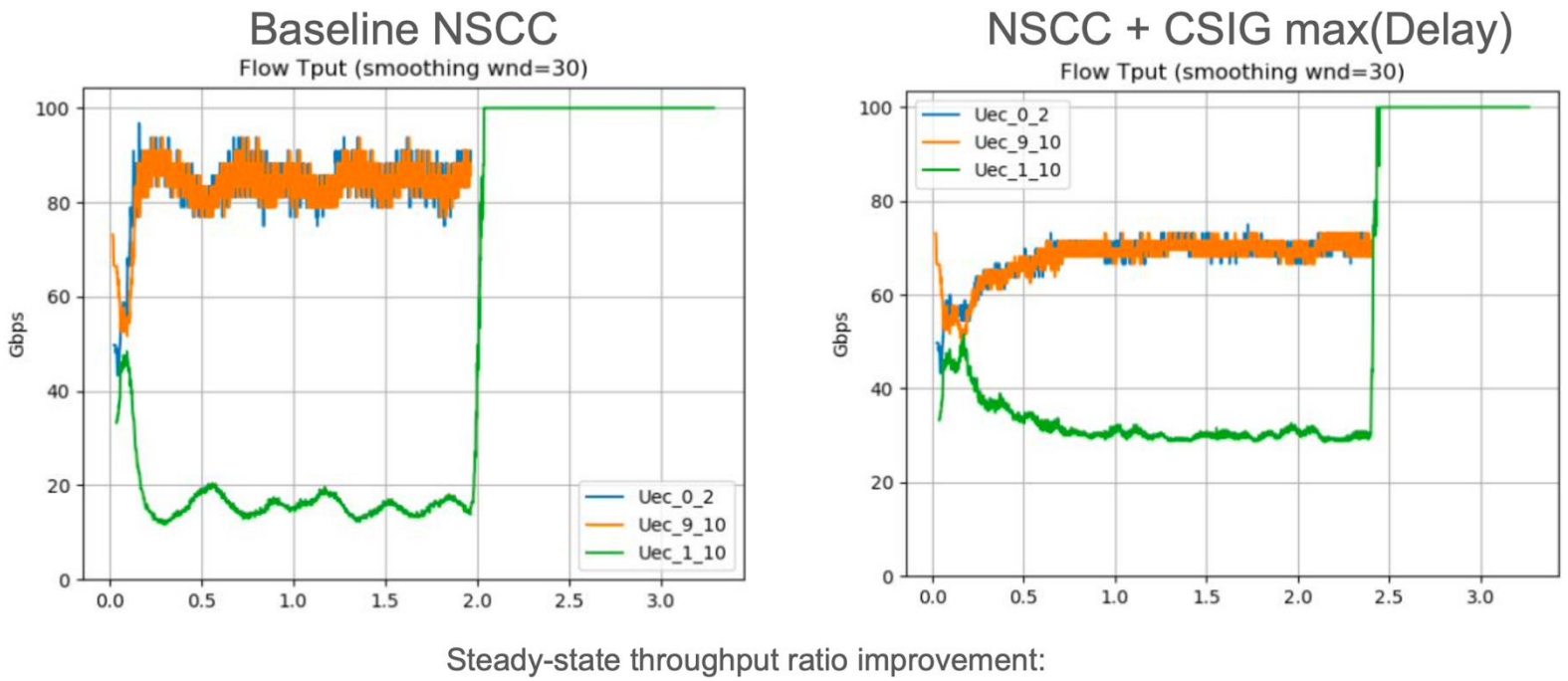
Bottleneck Sharing in Delay Based CC

- 3 Flow Scenario: Using end to end delay can result in unfair sharing when competing flows traverse different numbers of congested hops
- Victim flow 1 – 10 encounters congestion at TOR-0 (uplink) and Agg-2 (downlink)
- Expected delay-based CC behavior – Victim's flow observes two queues worth of delay but other two flows observe one queue's worth of delay
  - status-quo delay-based CC will throttle victim more severely than other two [Poseidon, NSDI 2023]



BROADCOM

Fairness Improvment

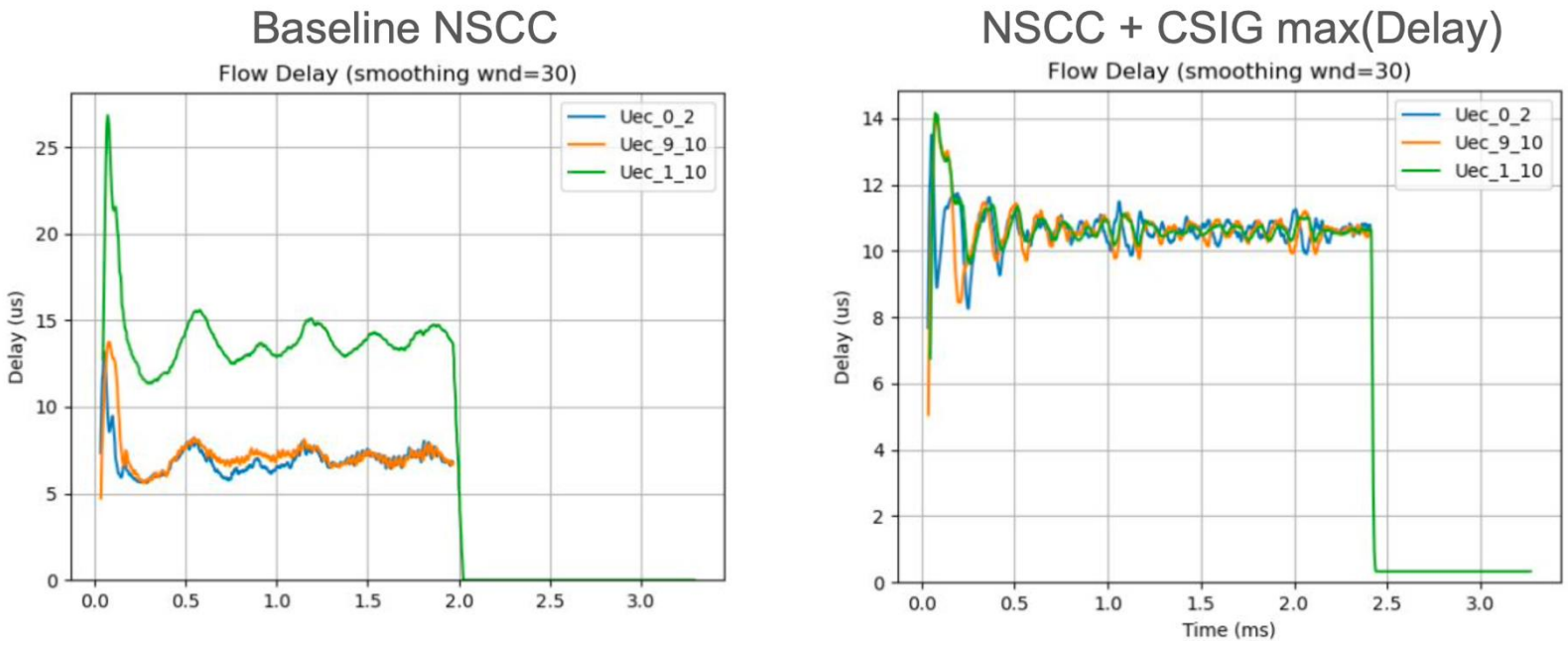


Steady-state throughput ratio improvement:

From ~5:1 (baseline) to ~2.3:1 (with CSIG max(Delay)

BROADCOM

Path Delay Fairness



target_delay = 10.5us

- Under baseline NSCC, victim flow's queueing delay remains persistently above target, whileother flows' queueing delays remain persistently below it.
- All 3 flows converge to a fair path delay

Thank You

BROADCOM®