Deterministic Networking (DetNet) Data Plane guaranteed Latency Based Forwarding (gLBF)

for bounded latency with low jitter and asynchronous forwarding in Deterministic Networks

draft-eckert-detnet-glbf-01

Toerless Eckert, Futurewei Technologies USA (tte@cs.fau.de),

Alexander Clemm, Futurewei Technologies USA (alex@futurewei.com),

Stewart Bryant, Independent (sb@stewartbryant.com),

Stefan Hommes, ZF Friedrichshafen AG (stefan.hommes@zf.de)

DetNet Interim meeting 07/19/2023, rev 1.0

https://github.com/toerless/detnet/tree/main/slides/interim-2023-07-19-glbf.pptx

Agenda

- Introduction
  - Positioning, History, Use-Cases
- Background
  - bursts, per-flow shaper, damper
- gLBF
  - Asynchronous Dampers, Dampers with UBS calculus
- High-Speed implementation options
- Validation
- Packet Metadata
- Benefits
- Summary

Introduction

Positioning, History, Use-cases

Positioning

- Gen 0: Imediate solution (small scale!) ???
  - DetNet via TSN: CQF, ATS (but operational complexity layering DetNet on TSN).
- Proposed 1st gen large-scale solution: TCQF / SCQF
  - Best currently possible match for large scale requirements (opinion)
  - Derived from long-term TSN used mechanism (CQF) – Deployment experience
  - High-Speed, Wide-Area validation (implementation, deployment)
  - Various options to work without new headers (TC, DSCP, SID)
- Proposed 2nd gen solution: gLBF
  - Incremental and integrative
    - Keeps TCQF/SCQF benefits
    - Adds benefits from TSN-ATS
  - Gaps/Work:
    - Depends on new packet header field(s)
    - No high-speed implementation validation
      - (When) will ASIC be able to do it ?!
    - No Formally/independently validated calculus
      - Specifically for high-speed implementation proposals

History / why

- Research work: “Latency Based Forwarding” (LBF)
  - Track latency hop-by-hop
  - Signal Min/Max end-to-end latency objective in packet metadata
  - Distribute per-hop propagation latency via IGP
  - Calculate available queuing latency == urgency of packet re. Competing packets
  - Prioritize packet based on urgency
- Did not manage (so far) to define calculus for admission control (too complex)
  - But likely very useful in conjunction with Congestion Control
- Investigated if/how principles could be applied to well-known calculus (UBS / TSN-ATS)
- Did no know that core mechanism was called Damper 30 years ago...

Use-cases

- DetNet for large-scale networks
  - Potentially also intersting for smale-scale networks
- Cloud PLC
- Remote Driving, machine/robot control
- “Outsourcing to cloud” of mobile devices compute (Cars, drones, ...)
- “Control as a Service”
  - Generalization, network and cloud-compute as integrated service for the above and other use-cases

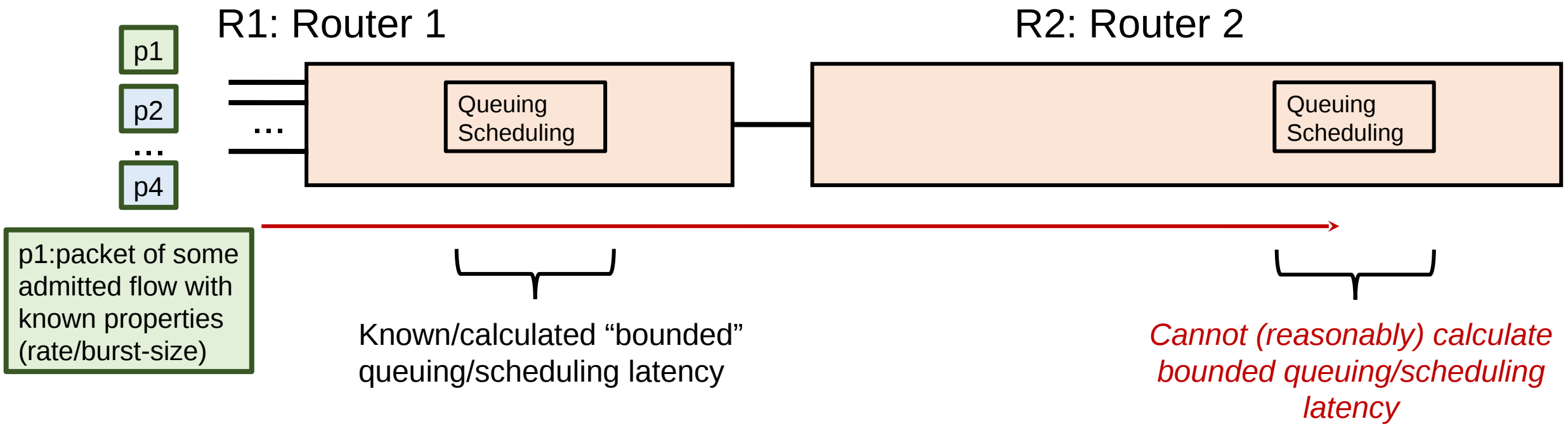
Tactile Operator (Client)

Actuators & Sensors (Server)

Background

bursts, per-flow shaper, damper

Guaranteed Latency, the Problem: Bursts



Pre: We have a model to calculate maximum bounded latency for packets of flows with known/defined properties across a router, e.g.: R1. We can therefore calculate how much buffer scheduling/queuing needs in R1 and when to stop admitting more flows because they would not fit anymore.

Problem: But we can not apply the same calculus for the following hops, e.g.: R2.

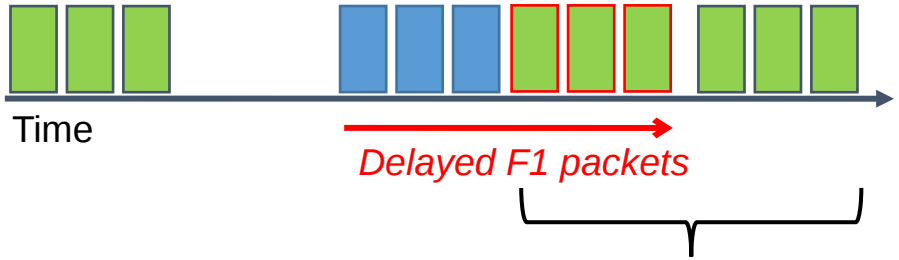
Reason: R1 receives packets from multiple interfaces all going to R2 (or from faster interface(s) than to R2)

Burst in R1

Flow **F1** (green) of interest has periodic burst of e.g.: three packets

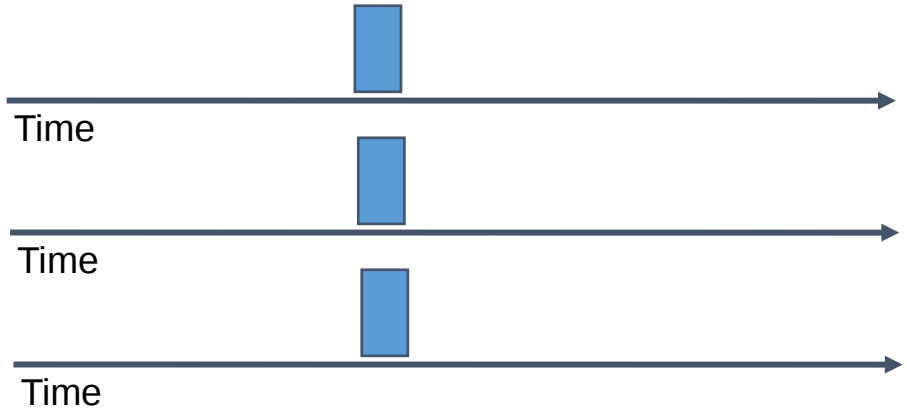
Before R1 queuing/shaping:

After R1 queuing/shaping: *(*)*



F1 Excess Burst (doubled)

Competing traffic flows on R1 from other input interface



On further hops, F1 now has for this period in time double the burst rate that it should have. This will invalidate any bounded latency admission control calculations

Example Bounded latency traffic model

UBS: if queuing/scheduling is just a single FIFO queue (in R1, R2, ...)

Bandwidth admission control of flows *(i)*

r(i) [bits/sec] **(guaranteed) bitrate of a flow**

Controller: Reserves sum of flow bitrates for each link

Bursts

b(i) [bits] **“burst size”**

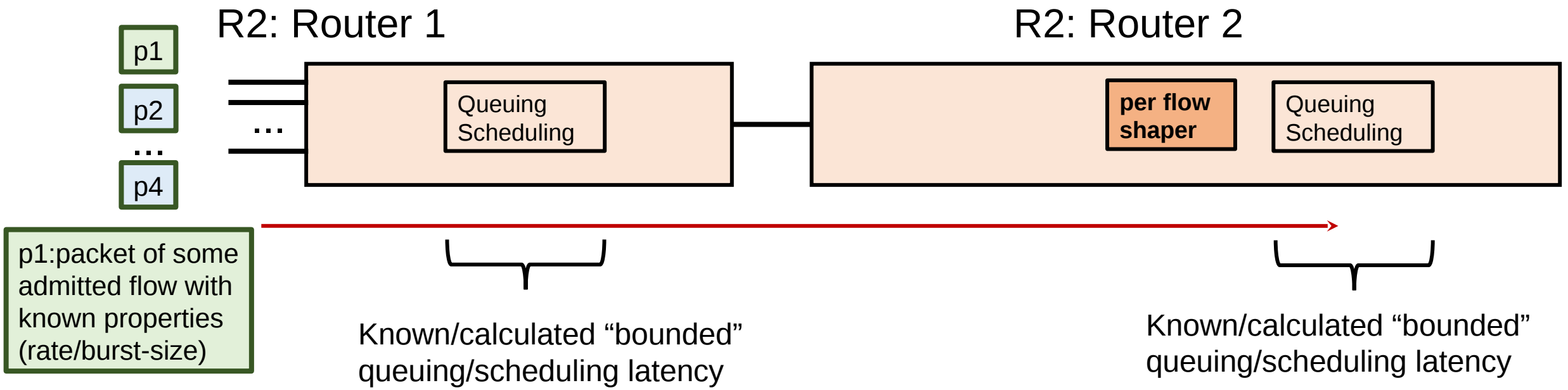
$w(i, d) \leq b(i) + d * r(i)$ “Leaky Bucket Condition”

Maximum per-hop queuing latency

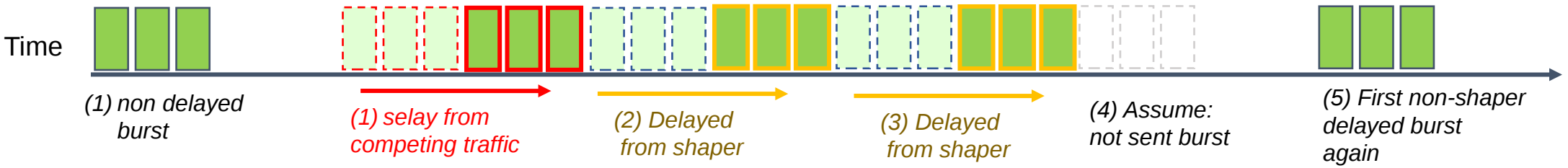
Buffer size needed on an outgoing interface

Maximum latency of packet

Classical Solution: per-flow shaper
RFC2212, UBS / TSN-ATS(interleaved regulator)



Shaper: when burst gets too big: delay further packets so allowed burst is never exceeded

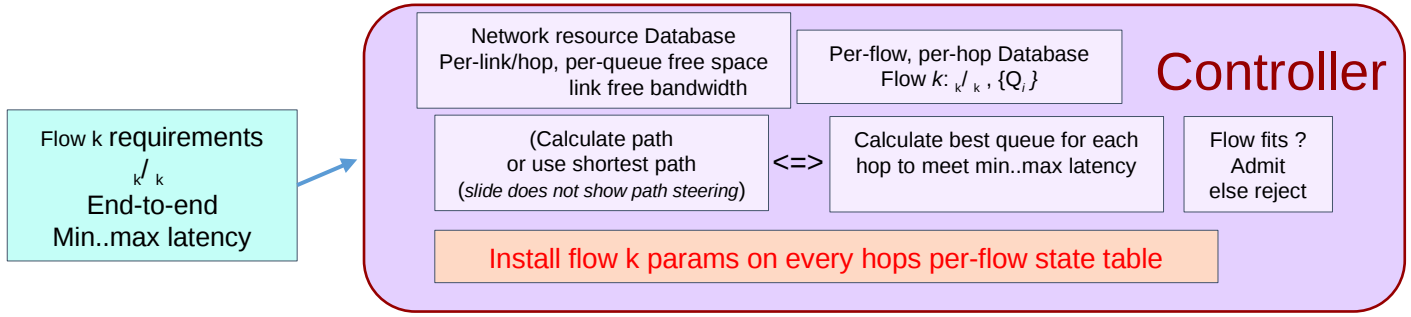


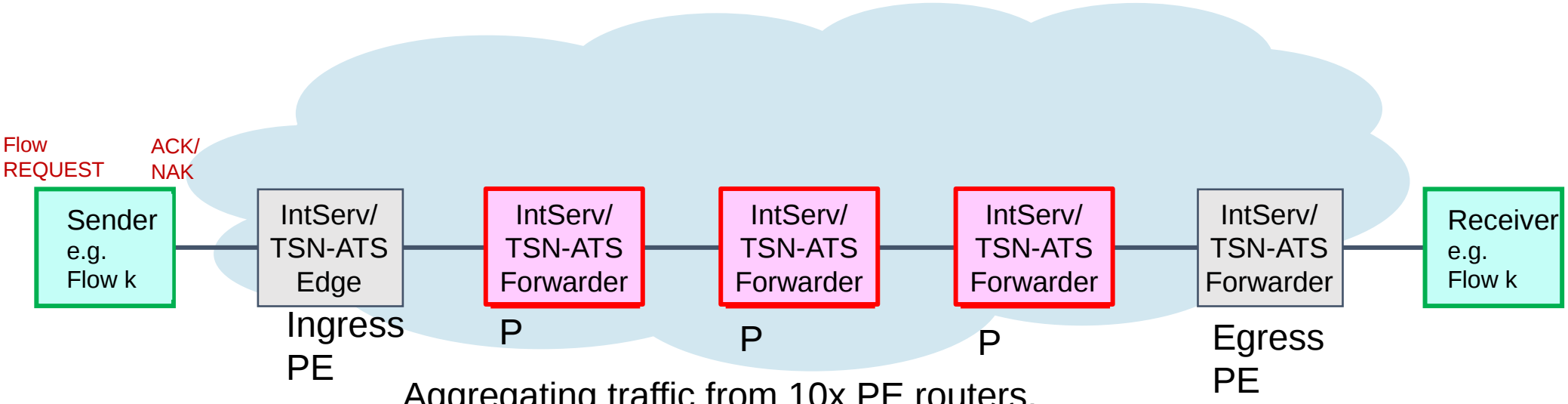
Why we do not like per-hop, per-flow bounded latency state

In large-scale DetNets (small, slow networks are fine ?!)

- Provisioning: every router hop needs to be provisioned with every flow keys and that flows shaper parameter
  - Violates P/PE design paradigm of large-scale networks (troubleshooping complexity, ...)
  - Creates lot of churn on control plane in dynamic deployment (flows come and go, network topologies change).
  - CANNOT use “source-routing” to eliminate them: shaper REMEMBER locally their state across packets
- Router performance challenge (expensive)
  - Shaper-parameter read/calculate/write cycles: Need to have for large number of flows high-speed write memory so shaper state can get updated before next packet for same flow is processed (high speed read/write cycles).
- These issues apply to interleaved regulators. Shapers do have on top also require the expensive per-flow scheduling cost at high-speed. Interleaved regulators (only) optimized that problem away.

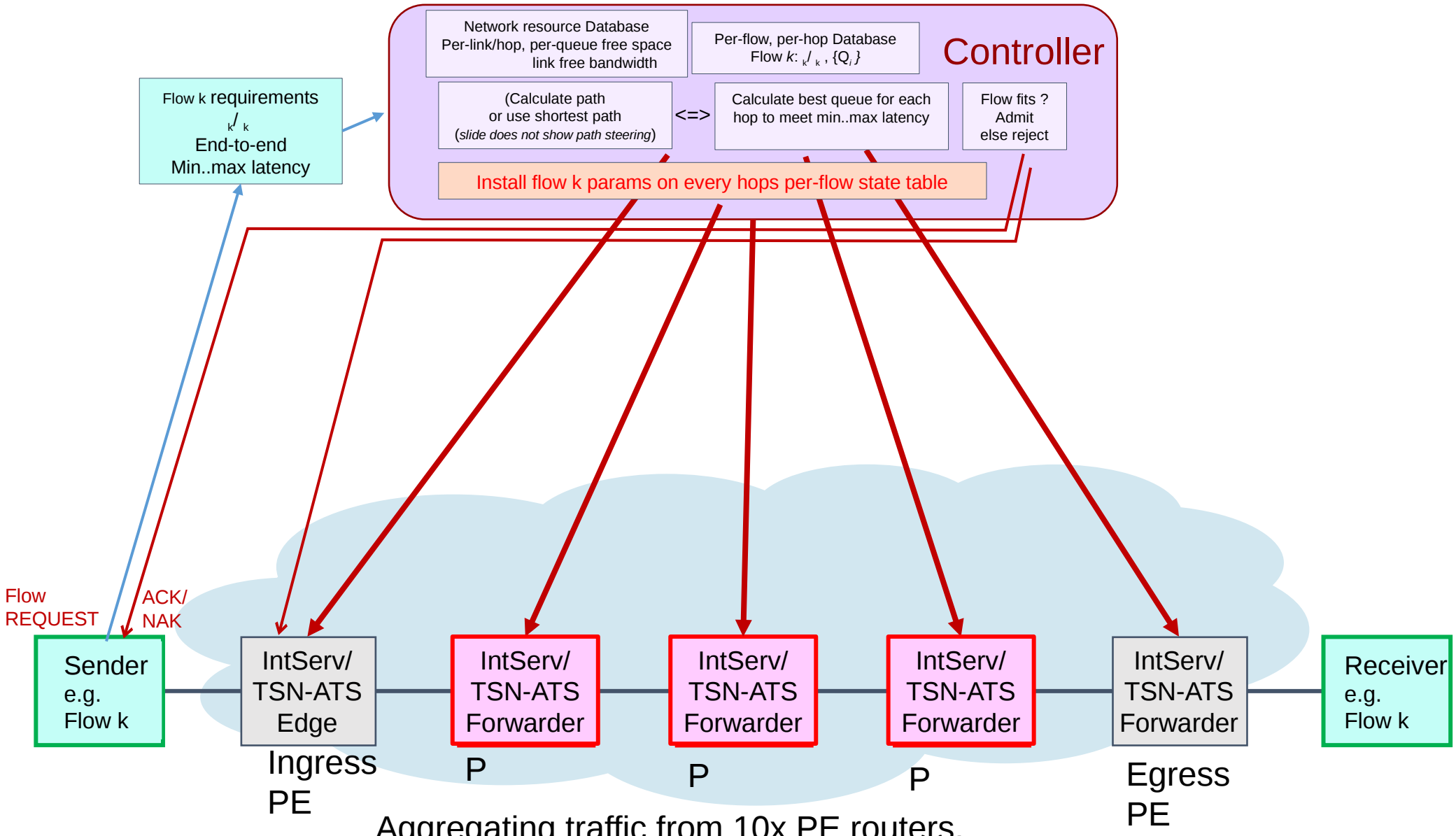
System model with RFC2212 / TSN-ATS





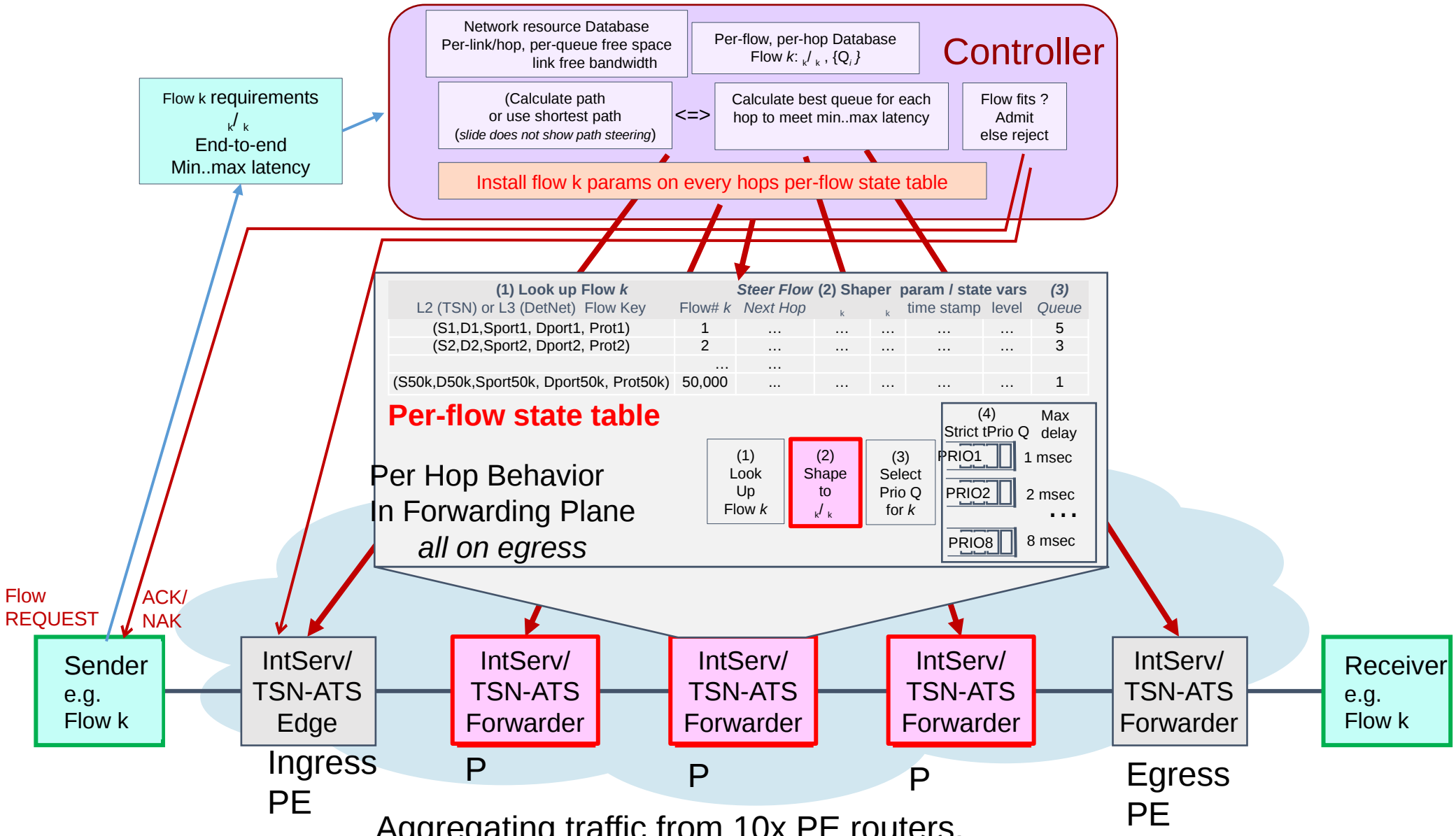
Aggregating traffic from 10x PE routers,
Very fast, cost/operational-sensitive (no-per-flow control wanted)

System model with RFC2212/TSN-ATS



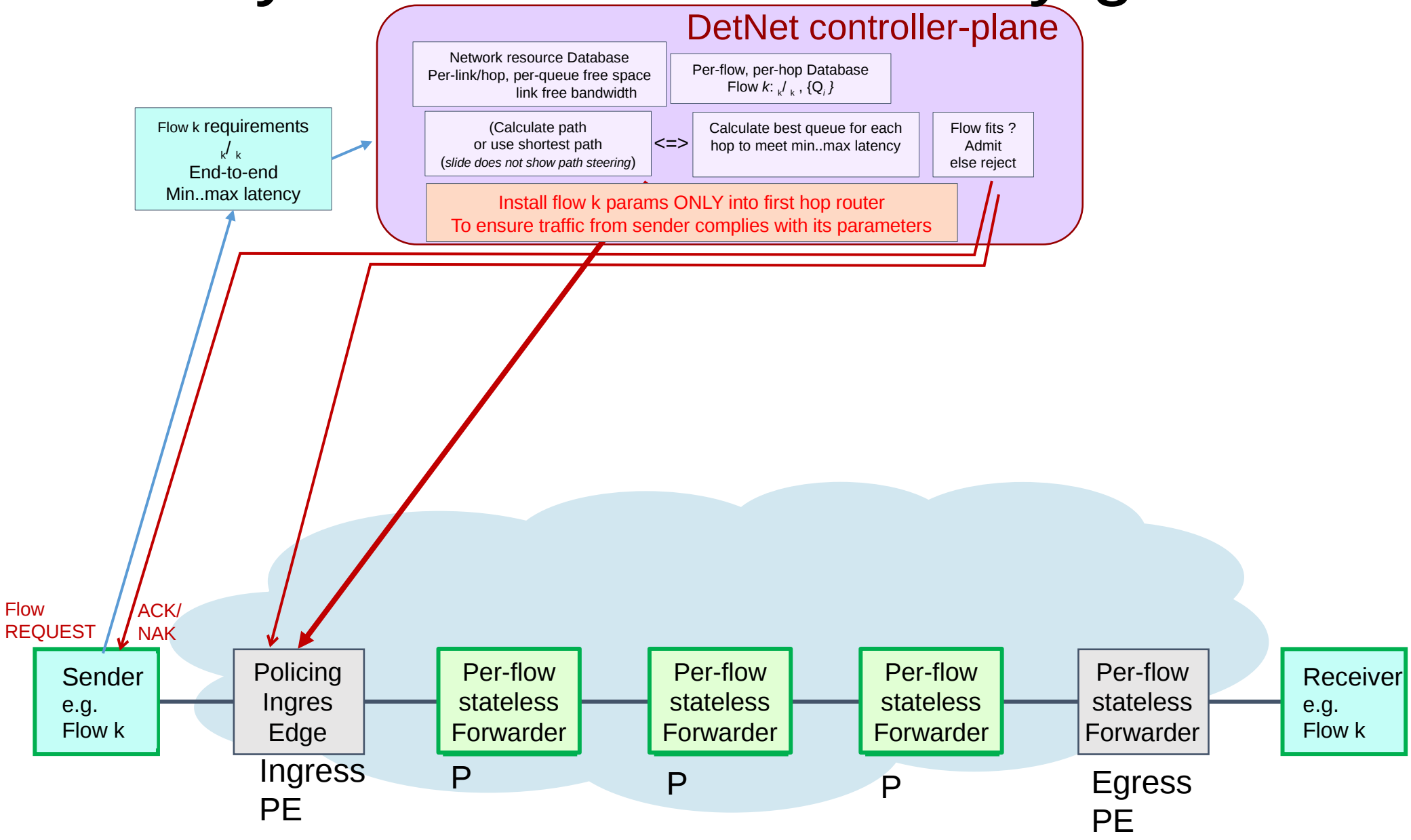
Aggregating traffic from 10x PE routers,
Very fast, cost/operational-sensitive (no-per-flow control wanted)

System model with RFC2212/TSN-ATS

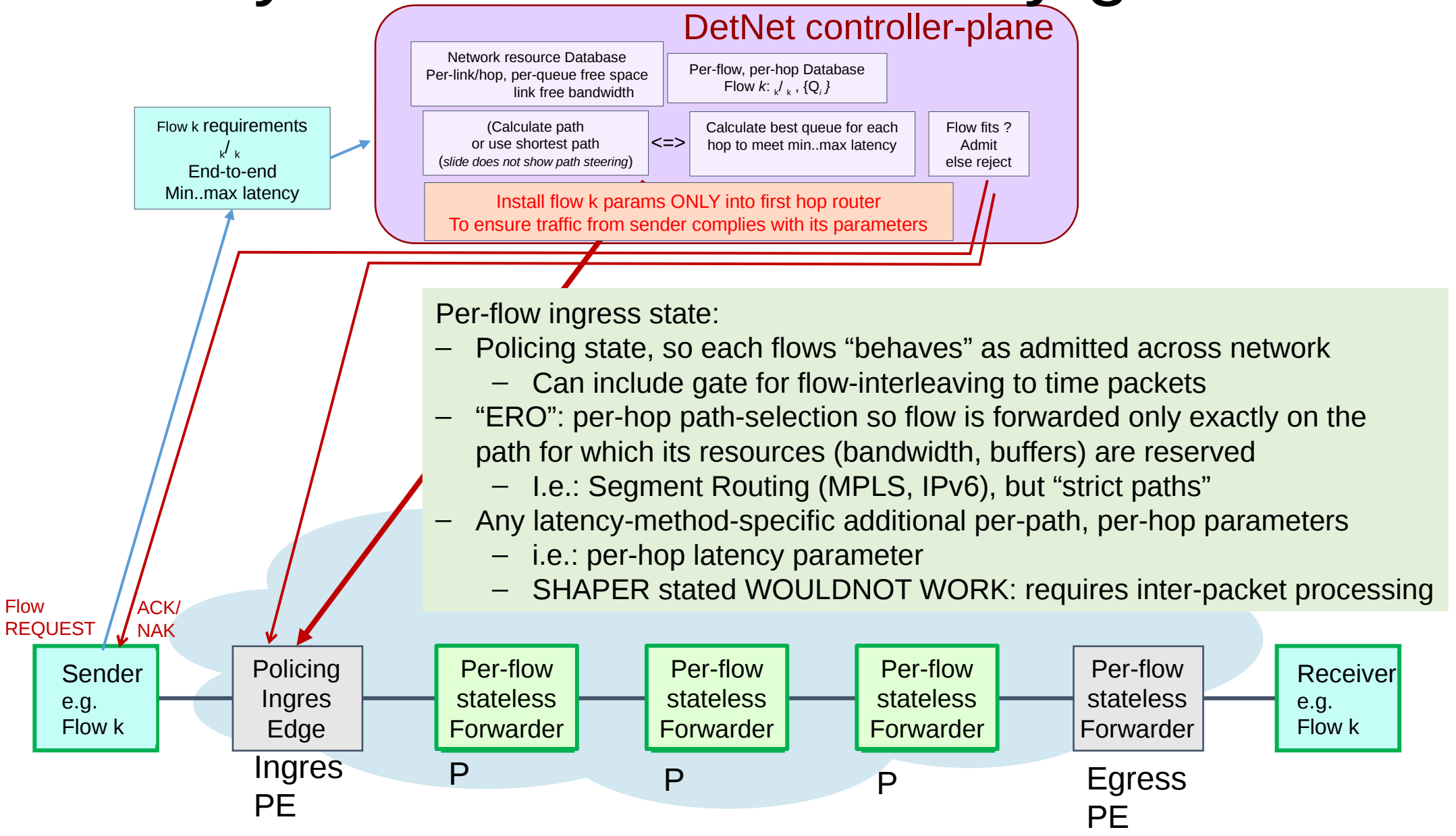


Aggregating traffic from 10x PE routers,
Very fast, cost/operational-sensitive (no-per-flow control wanted)

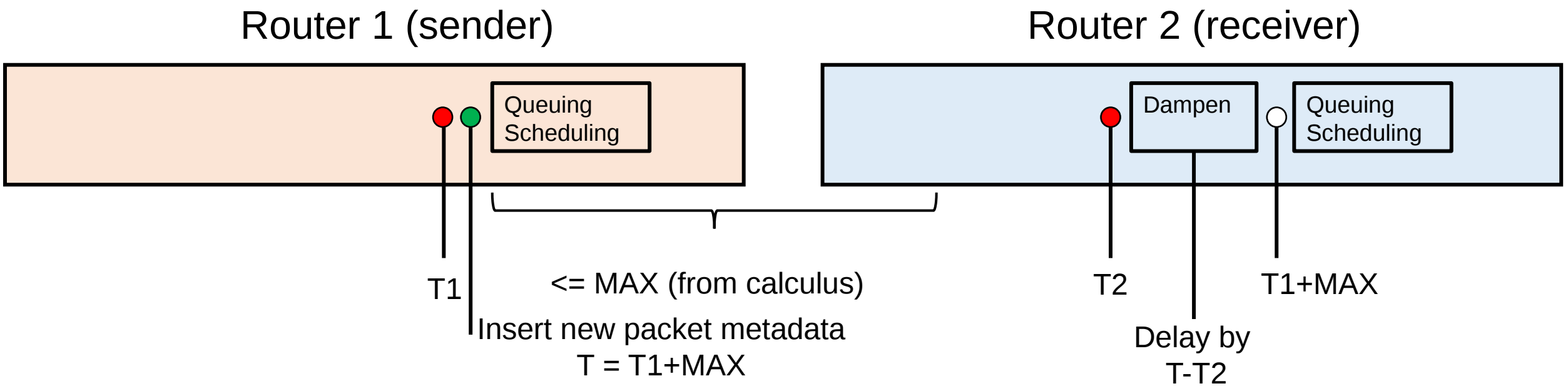
Desirable system model of latency guarantee



Desirable system model of latency guarantee



Dampers – basic concept

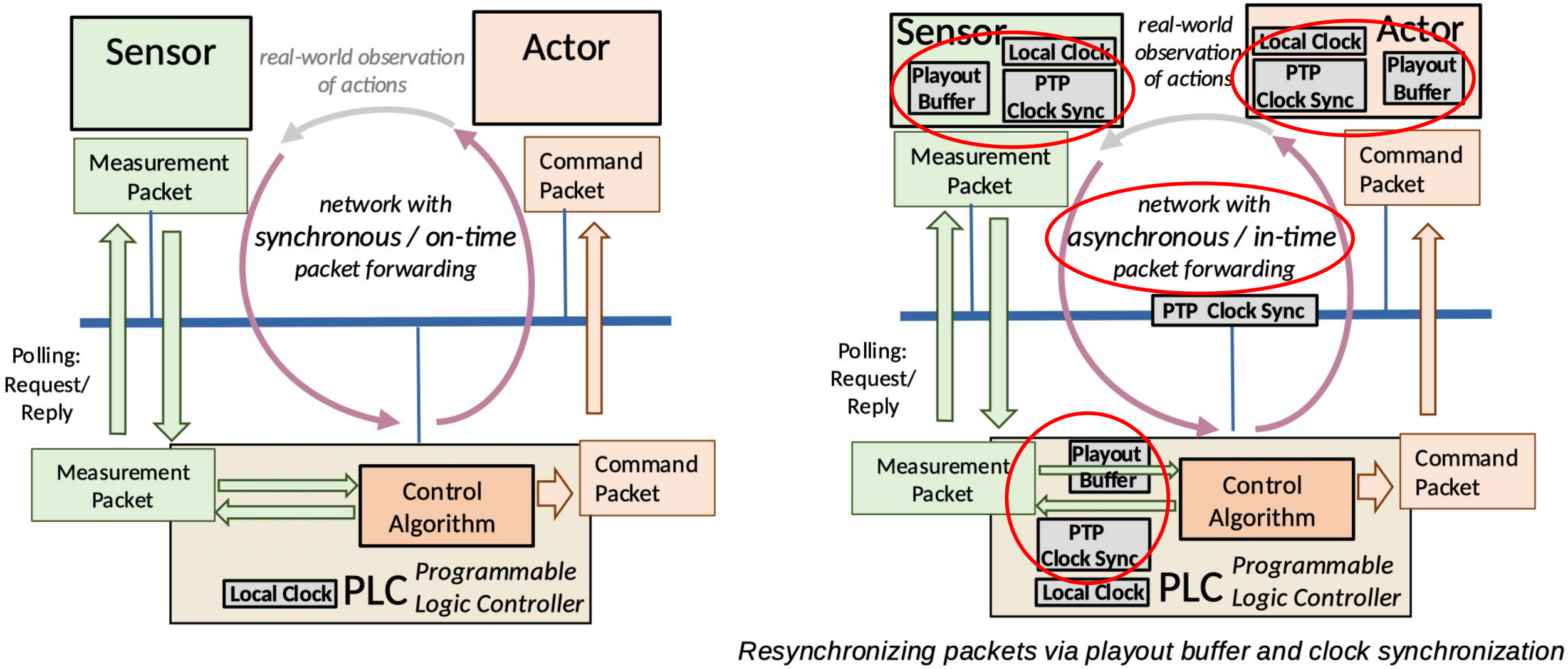


- MAX calculus needs to include not only queuing/scheduling latency but also link-transmission (details later)
- With Damper, all packets will have exactly the same inter-arrival time at R2 queuing/scheduling as they did on R1
  - We simply always delay all packets per-hop to their maximum guaranteed latency
  - This eliminates impact of any bursts on R1
- If we could calculate MAX on R1, then so can we on R2. We eliminated burst impacts.
- And no per-hop, per-flow state required

But wait...

- With RFC2212/UBS/TSN-ATS we only delay packets when necessary (during bursts and as needed after wards). Most times, most packets will be delivered much earlier than maximum latency for this hop. With Damper, they will always be maximum “late”.
- Yes, but this is actually beneficial for many if not most (stringent) applications
  - Synchronous delivery, control loops / media playout
  - Eliminates need of receivers to delay packets to make them synchronous.
- Eliminates need for many application components (actors/sensors) to have local synchronized clocks
  - Good enough for one side of communications (e.g.: PLC) to have clock. Arrival/send time on other side known because latency is synchronous/fixed.
- Shorter latency never guaranteed. Often dangerous to rely on it!
  - If you run control loops faster than with guaranteed latency, it may easily crash when latency goes higher (but stays within bounded latency of course!).

Benefits of synchronous packet transmission



Resynchronizing packets via playout buffer and clock synchronization

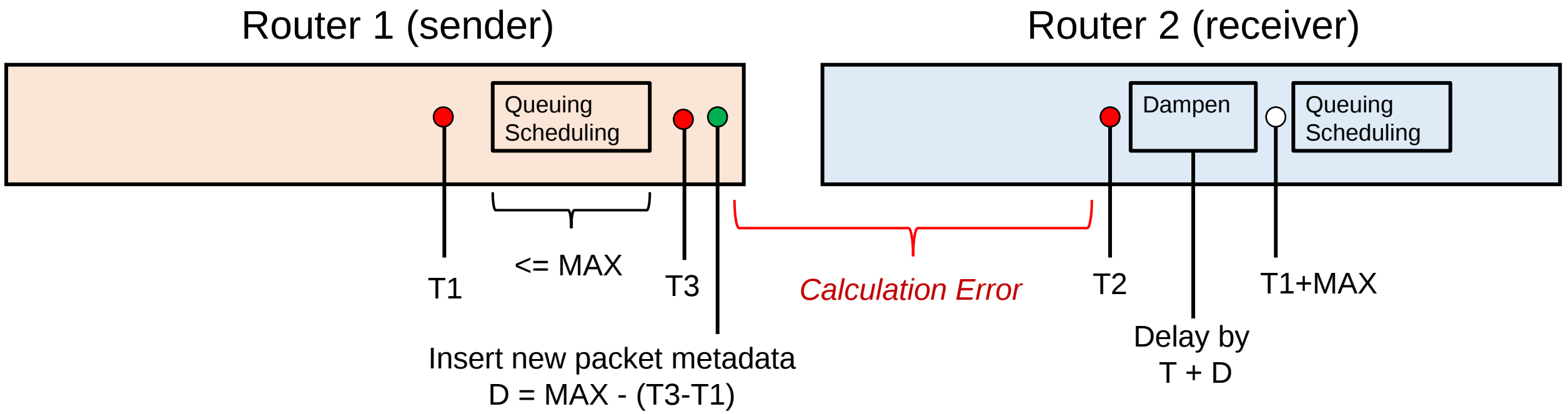
gLBF

Asynchronous Dampers, Dampers with UBS calculus

But wait (2)...

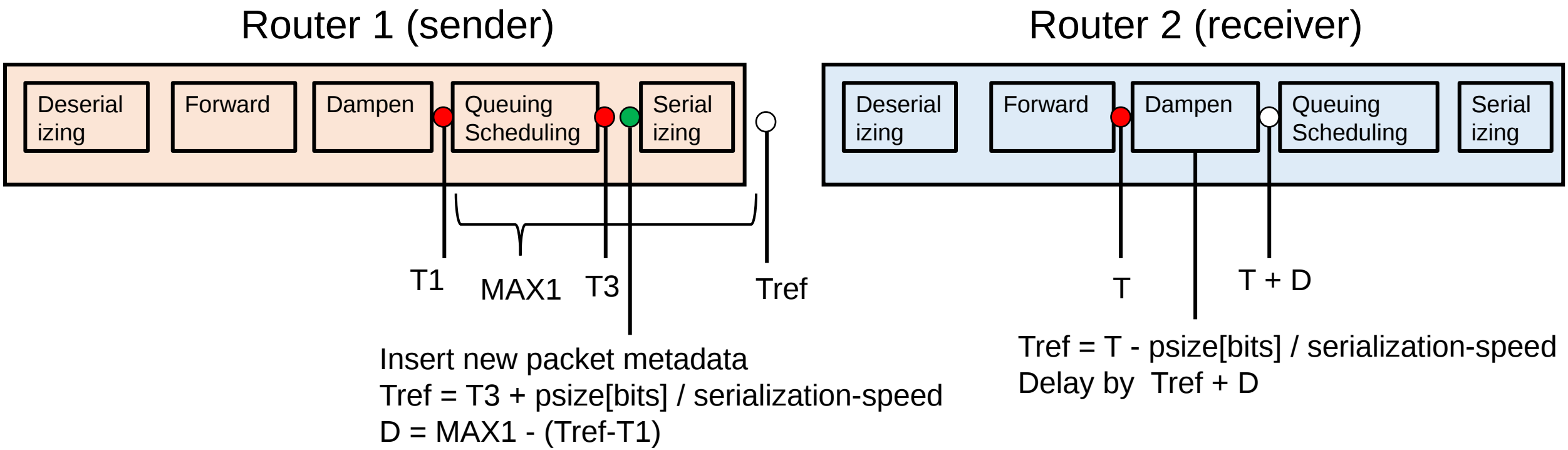
- Dampers introduce clock synchronization needs between R1/R2, aka: across network.
- Clock Synchronization is highly undesirable in most large-scale networks
  - One of the reasons, why CQF/TCQF/SQCF may not be ideal

Asynchronous Damper – basic concept



- Now we transmit no (target delay) timestamp, but a time-delta as metadata: D.
- Eliminates clock synchronization need between R1 / R2
- But this is incomplete: The time between R1:T3 and T2 is not considered
  - Serialization latency: proportional to packet length – aka varies across packets. MUST BE CONSIDERED
  - Other VARIABLE delay: in-router, or network transmission (radio link, l2 retransmission, reflections,...)

Asynchronous Dampers: gLBF damping



- Tref is time of first bit of packet (TBD: L3 or L2/L2) observable on egress of R1
- MAX1 is MAX + serialization speed of largest packet allowed
- Assumes same serialization/deserialization speed (simple, common case)
  - “easily” adjusted if wired link properties are more complex (subrating etc..)
- Does NOT consider speed of light propagation of link: NOT VARIABLE (enough) to care!

gLBF so far

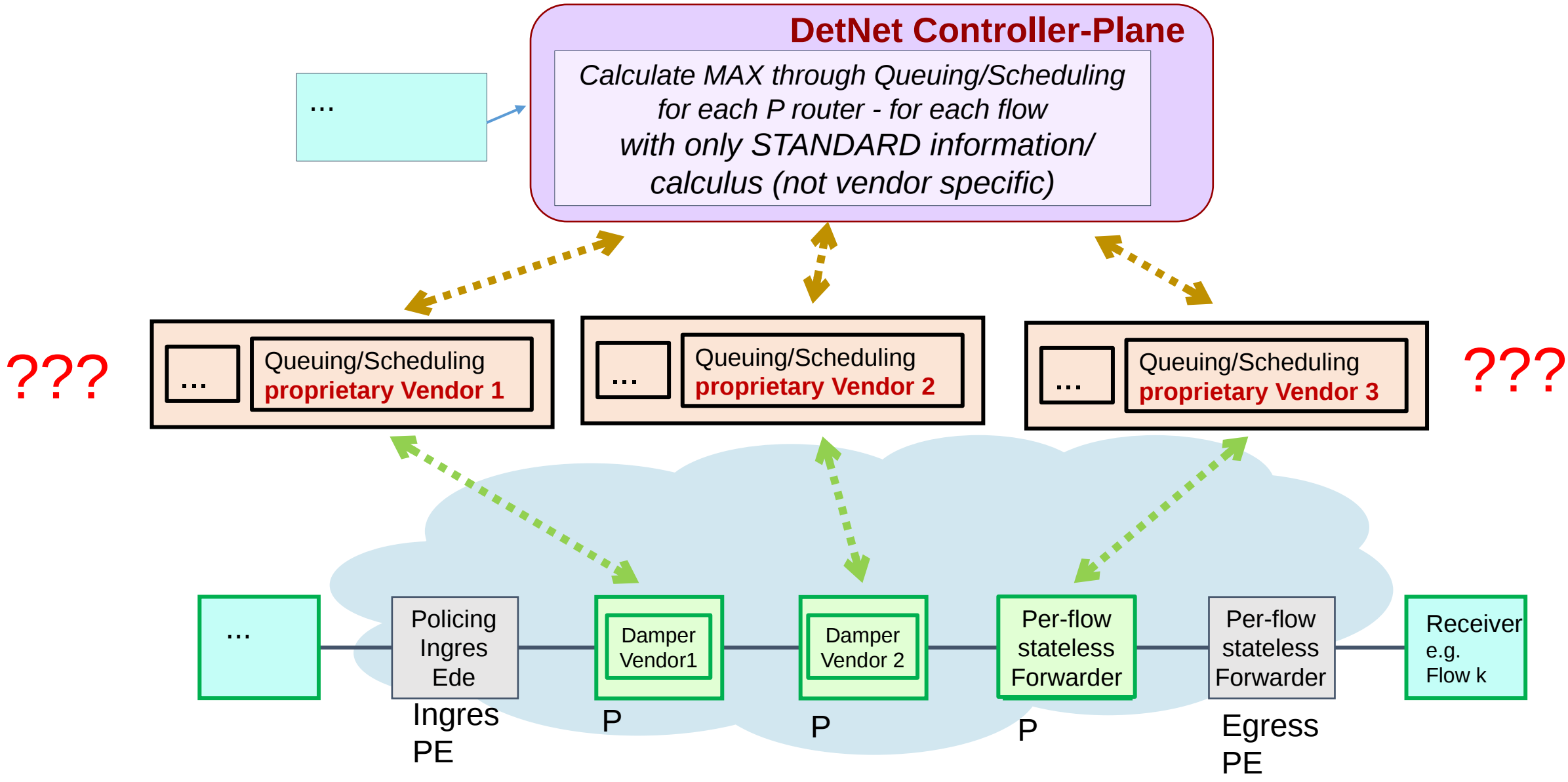
- We’re not (yet) at actual high-speed implementation proposal!
- For wireline links: use asynchronous gLBF model
  - No clock synchronization needed
  - Metadata: D(amper) value.
    - E.g.: nsec resolution, >= 20 bit (100 usec max hop latency)
- For edge-radio links (optional – RAW problem)
  - Use synchronous model (picture details not shown – left as exercise for readers)
  - Clock synchronization for radio links acceptable ?!
  - Metadata:Tref

  - Allows to compensate for variability of radio link
    - Retransmission at L1/L2, delay by reflections at building, speed changes by modulation...
    - See RAW architecture

Great ? Done ?

- Yes, Great, But not Done!
- This is not complete enough for standardization
- This is not complete enough for high-speed implementation

Standard system requirement



Standardization Requirement

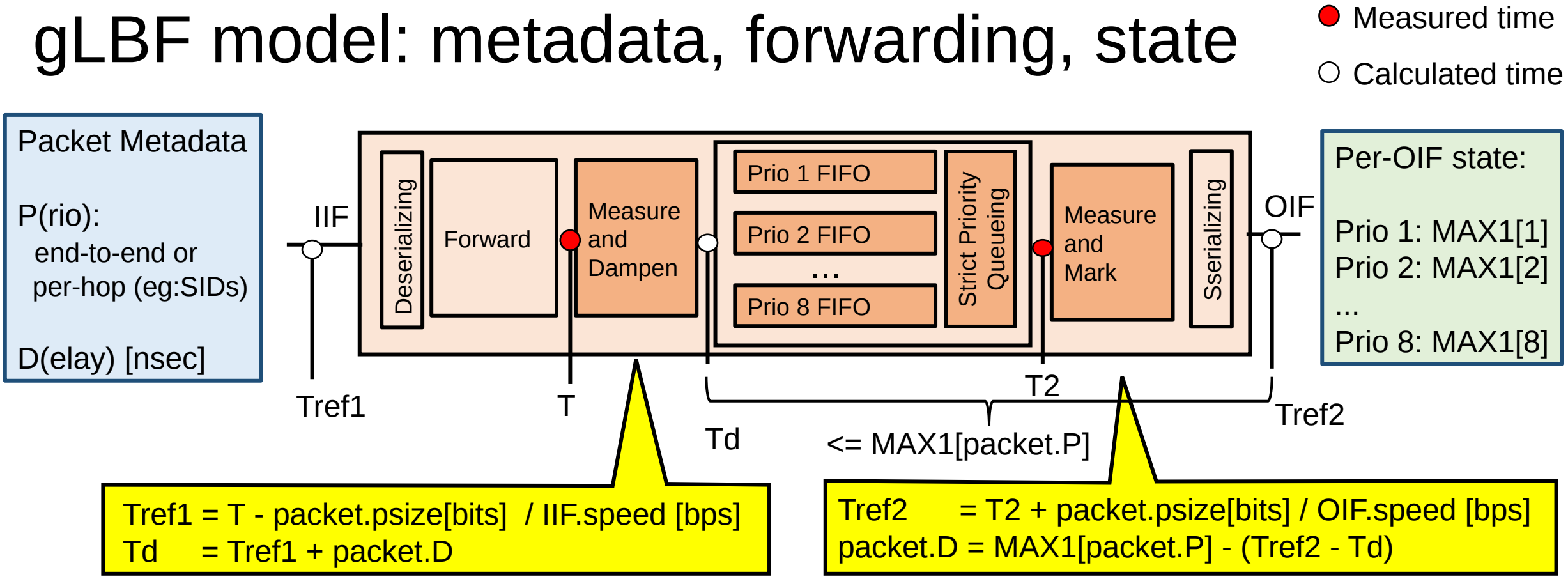
- Need to pick (one or more) standard Queuing/Scheduling mechanism to use in conjunction with Dampers
  - gLBF proposes to use UBS / TSN-ATS Priority Queues

- Or else it will me impossible to build good, vendor independent controller-plane. Proof: left as exercise to readers!

- Consider that controller has potentially complex “best-latency/throughput path optimization” problem to solve

- Solutions with vendor-specific proprietary controller-plane plugins would limit flexibility of algorithms in vendor independent controller-plane, eliminate many optimizations based on knowledge of common, standardized per-hop latency calculus.

gLBF model: metadata, forwarding, state



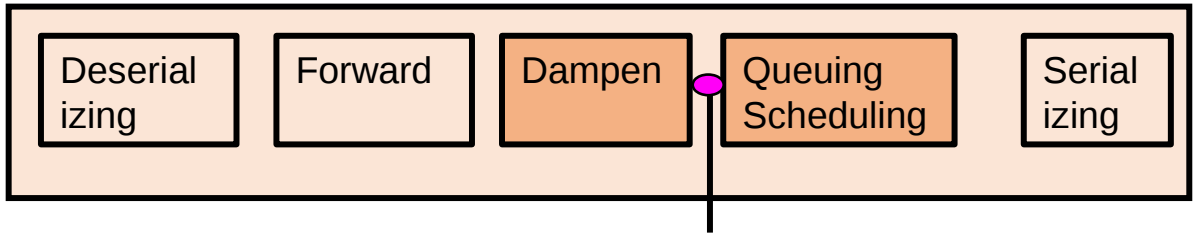
- Assumes forward step to have no relevant time variation to allow T measurement to be taken after it (egres linecard). If relevant, move T before forward and include in calculations.
- Calculating Td instead of measuring it means Dampen and Queuing/Scheduling can be single-staged in implementation.

Proposed High Speed (ASIC/FPGA) implementation options

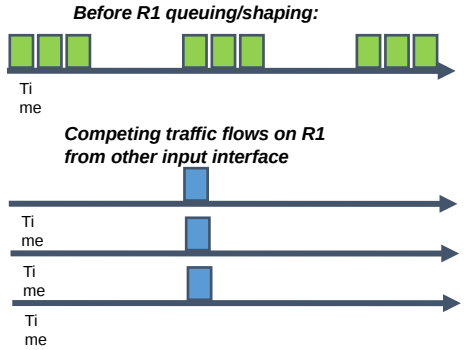
Internal behavior – not part of standard (appendix in draft).

Not implemented, not validated!

Performance requirement



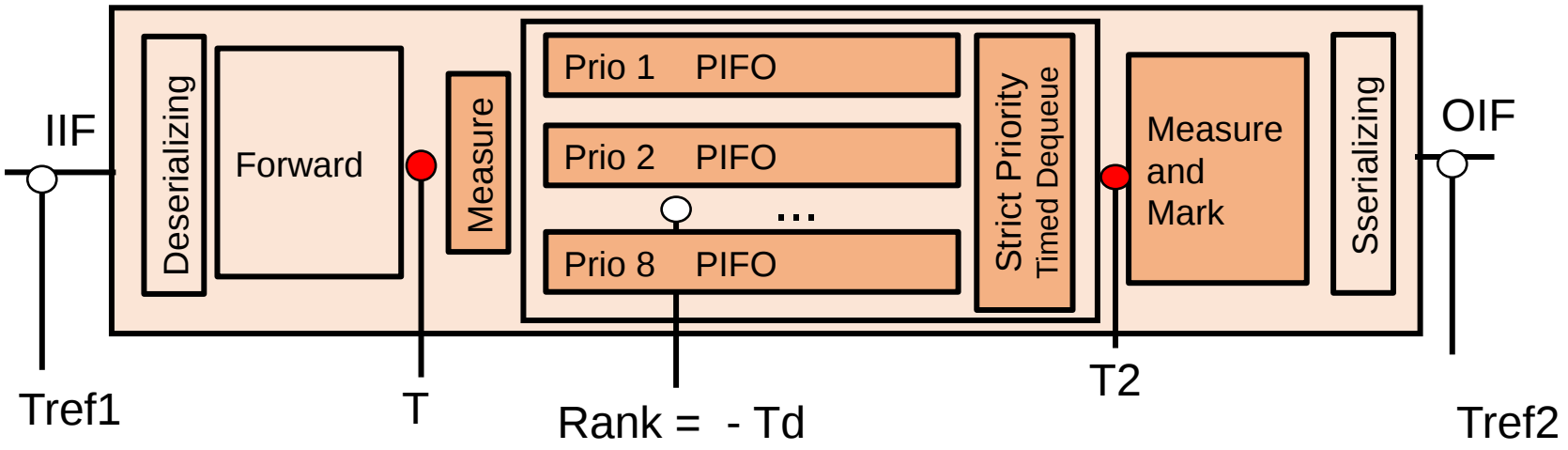
Performance Bottleneck point



Prior burst example

- Dual stage damper / queuing/scheduling is highly undesirable, maybe even impossible at high speed
- Remember burst example: packets arriving at same time, building queue.
- With dampers, these packets would be simultaneous again next hop after the Damper (before queuing)
- If applications send from different places in network packets simultaneous, then worst case the number of ideally simultaneous packets at the bottleneck point could be factor 100 or larger
- Impossible to build processing much faster than without Damper – without big aded cost
- Sequential processing only feasible at low speed or by introducing (high speed) big complexities: dequeuing from Damper, accounting for added dequeuing/enqueing latency to Queuing/Scheduling stage
- Really need single-stage solution for high-speed
- This requires well-defined/standardized Queuing/scheduling stage.
- Luckily we already need this for system standards requirement

gLBF high speed option 1: timed PIFO

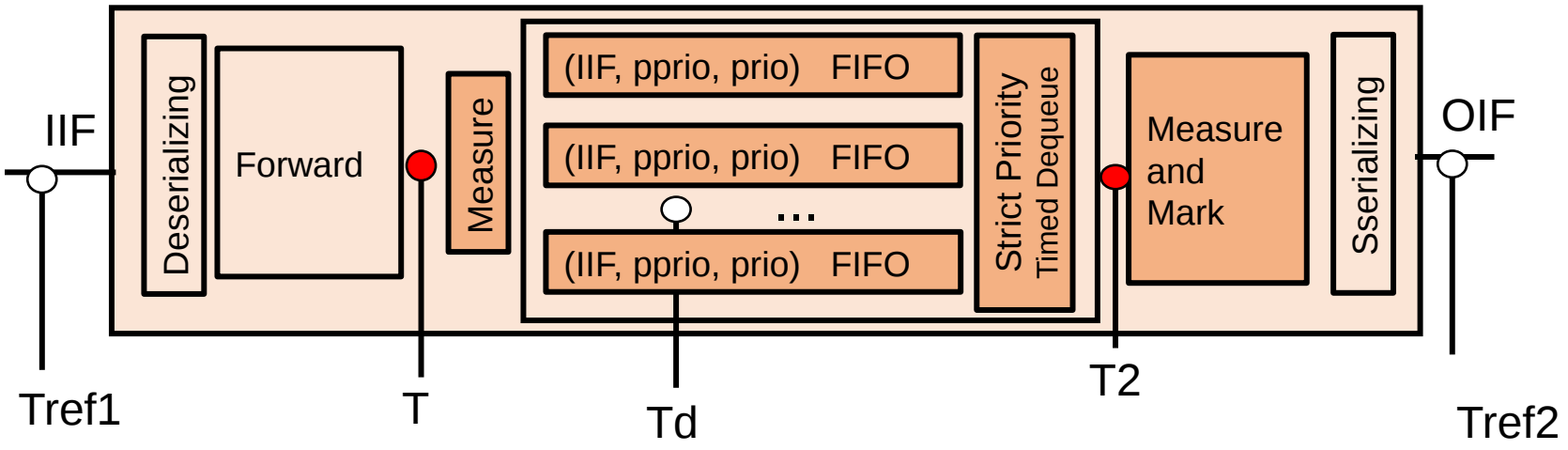


- Unchanged calculations, semantic of behavior. Removed Damper stage
- FIFO stage replaced by Push In First Out (PIFO) with timed dequeuing (timed PIFO):
  - Each packet in PIFO has a Rank. Packets are inserted into PIFO queue in order of decreasing Rank. Highest Rank is Top-of-PIFO.
- In gLBF, Td is target damper time: Time when packet could first be sent. be sent (absent packets from a higher priority PIFO). The earlier Td, the higher the Rank of packet. E.g.: Rank = –Td.
- Head of each PIFO can first be dequeued when time reaches (negative) Rank of Head of PIFO. Strict Priority dequeuing across the PIFOs.
- Effectively PIFO serves as both as Damper and as Priority FIFO stage: as long as TD is not reached for a packet, it is logically in the Damper stage, afterwards it is in the FIFO stage of processing.

gLBF high speed option 1: PIFO

- Promising research PoC implementation of High-Speed (100Gbps or more) PIFO in FPGA with scale better than O(#flows)
  - Early PIFO PoCs had scale/cost O(#flows) == O(#not-in-order-packets)
- Currently relying on fixed Rank though
- When timestamp is used as Rank (as gLBF needs), some additional logic is needed to wrap some fixed range of Ranks
  - 32 bit timestamp with nsec resolution would wrap ever 4.2 seconds.
  - Size of rank will impact cost of implementation
  - Range needed for ranks at at time is O(MAX1[8]).
- Unclear how difficult it is to add this missing piece in FPGA code

gLBF high speed option 2: timed FIFO



- PIFO can be replaced by FIFO – as long as packets are enqueued with increasing Td
- Need a separate PIFO for each combination (IIF, pprio, prio)
  - Pprio is priority of packet on prior hop. Prio is P(rio) in packet header
  - Packets arriving from same IIF with the same prio on this hop and same prio on prior hop arrive in the same order as their Td
  - *Addtl requirement: Needs pprio in packet header metadata*
- Same idea as high-speed implementation in UBS: per (IIF,prio) interleaved regulator
  - Except no shaper state/processing ; instead needs more FIFO

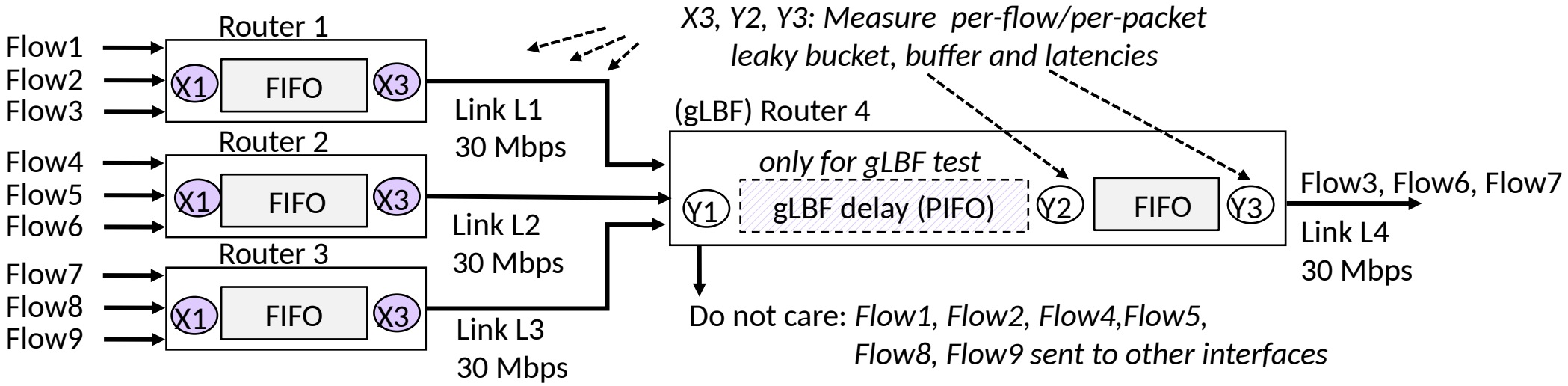
Discussion

- No per-flow lookups required. No register write (as in updates for shapers/interleaved regulators)
- Timed priority dequeue based on head of queue time (Td) same functionality as in UBS/TSN-ATS
  - In TSN-ATS this is calculated by shaper, in gLBF by delay calculation
  - Should hopefully be well feasible for high-speed, low-cost
- Per (prio,pprio,IIF) FIFO queues same concept as in UBS: Uses per (IIF,prio) queues to also ensures queue will only receive packets in order in which they will have to depart (no need insert into middle as in PIFO solution).

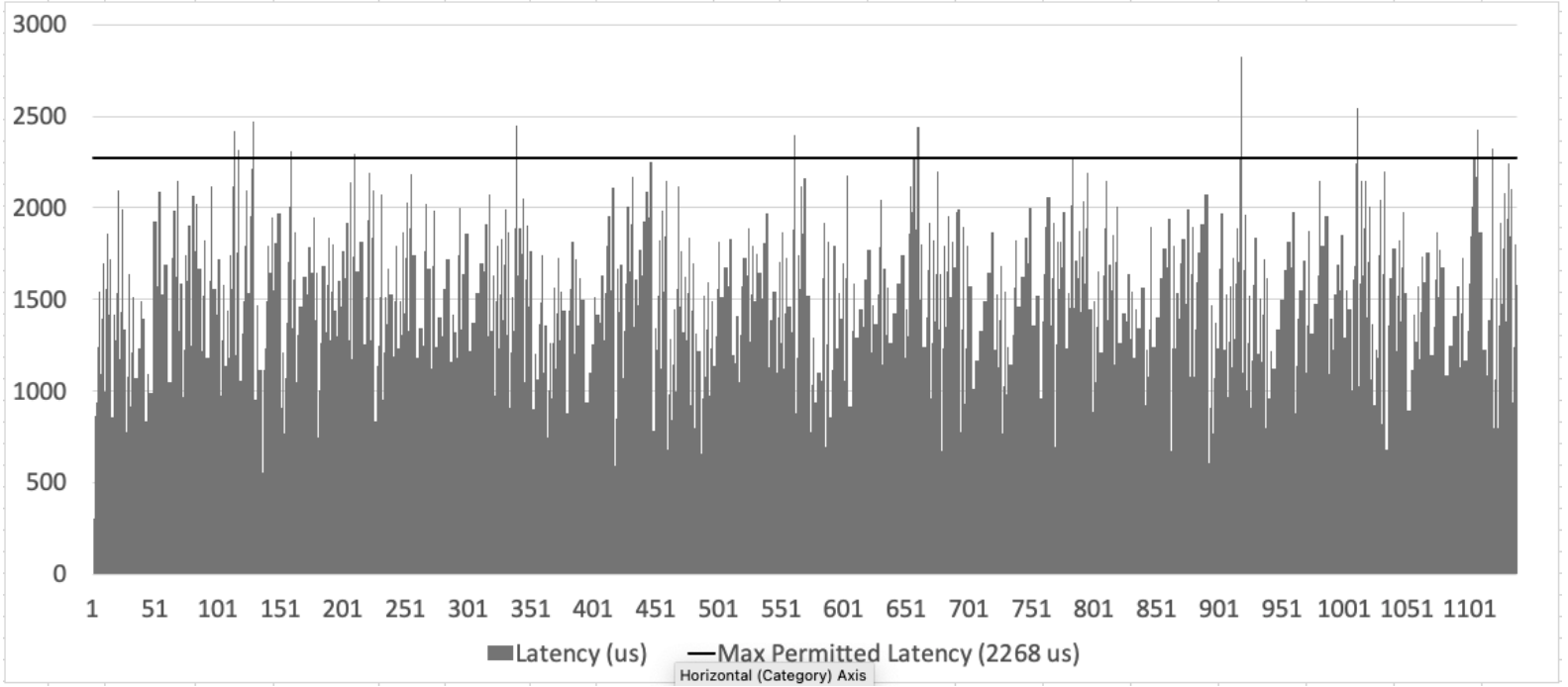
Validation

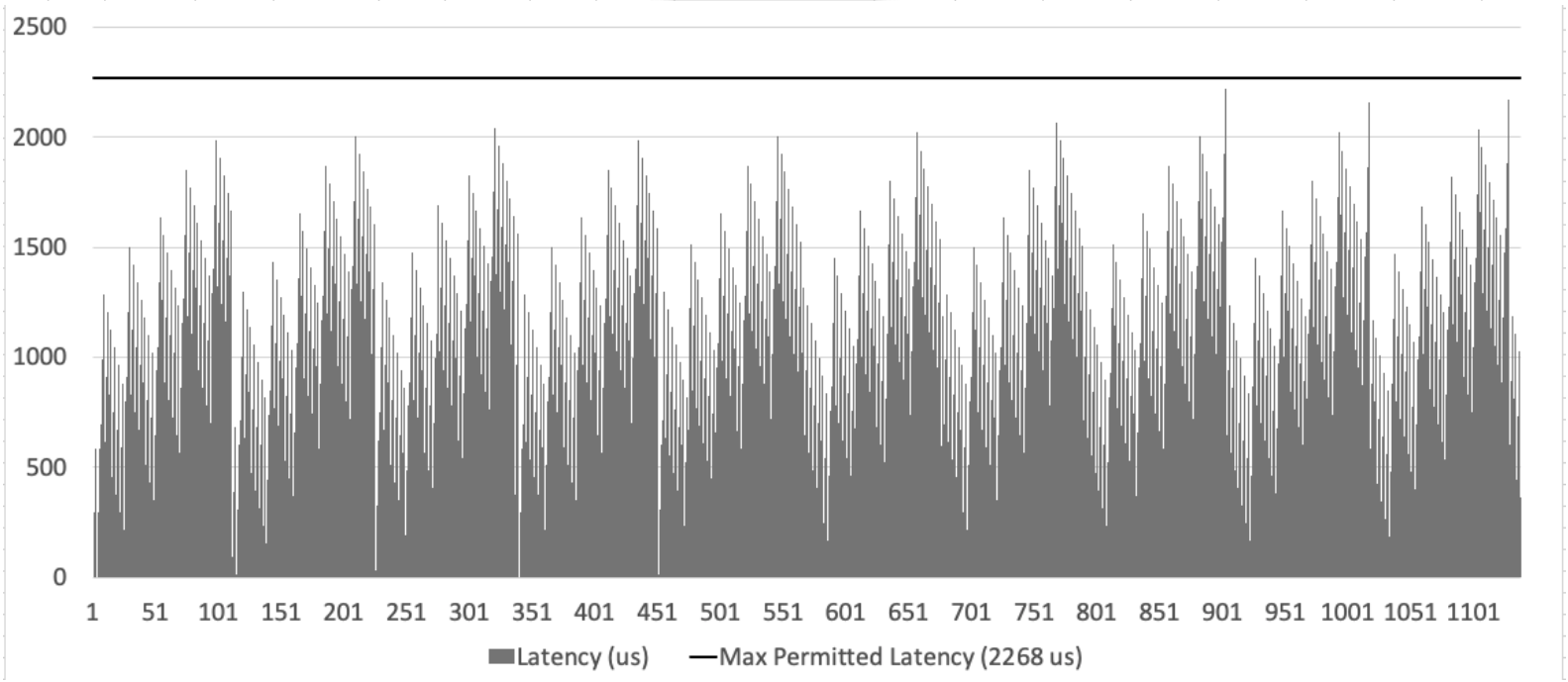
Validation

- Ad-hoc simulation implementation, 1 Gbps, 4 routers, 2 hops
  - 1st hop to generate burst collisions
  - 2nd hop with or without gLBF damper, compare results
- Result
  - Without damper, some flows would show higher than guaranteed latency across second hop
  - With gLBF damper (of course) no excess latency.



Validation





Packet metadata / encaps

Packet metadata overview

No actual proposals for encap, just various ideas...

- Some indication that gLBF processing is desired for packet
  - Could be presence of Damper or Priority field's in packet or TC/DSCP, or ...
- D)delay (Damper) : for hops using asynchronous mode
  - per-hop rewritten. >= 20 bits, suggested unit: nsec (1 msec max buffer ?)
  - MPLS: 2 top of stack labels, special function + damper value ??
    Or better post stack data (consistency with IP/IPv6 ?)
  - IP/IPv6: in common DetNet header
- Td: Target damper timestamp: for hops using synchronous mode
  - Could be same size as Damper field, e.g.: 20 bit, resolution nsec,
    synchronized clock timestamp modulo 1 msec
- P)riority: 3 bit for 8 priorities (to match TSN-ATS)
  - Always read-only, not changed in processing
  - Per-path priority: single metadata field
  - Per-hop priority, embed into SID based steering model
    Can use all the same ideas as CSQF
    - E.g.: 8 SID per OIF or 3 bit parameter (SRv6/SRH/RFC8986)

Benefits

Benefits re. CQF / TCQF / CSQF

- Less jitter than *QF cycle time
  - Depending on accuracy of processing of timestamps in router / metadata
- Can be asynchronous, no PTP clock sync required
- Can support difficult/jittery links (radio) – synchronized only across link

- TCQF backward compatible: Set up gLBF to behave like TCQF:
  - Single Priority, MAX1 buffer space same as TCQF cycle would be
- CSQF backward compatible: Simpler model for fine-grained per-hop latency management:
  - Set up 8 priorities, each with same buffer size. Achieves per-hop synchronous latency of T, 2*T, 3*T, ... 8*T.

- Higher flexibility
  - Buffer sizes for each hop == delay for each hop independently configurable. In TCQF, CSQF these need to have the same time

Benefits re. UBS / TSN-ATS

- Same traffic model, bandwidth/latency calculus as UBS / TSN-ATS
- Should be able to re-use most of existing TSN-ATS controller-plane
  - Specification of flow envelope
  - Algorithms for determining per-prio buffer-space, provisioning of buffer
  - Algorithms for path calculations for flows, calculating per-hop priority of flow, admitting flow
- Can be asynchronous, no PTP clock sync required
- Can support single (radio) link propagation variation requring only single link clock synchronization (but not path wide)
- Plug & play co-existance and/or replacement ?
- Core difference (benefit ?!):
  - On-time packet delivery (gLBF)
  - In-time delivery (UBS)

Summary

Summary

- gLBF is adoption of old Damper idea
  - Enables per-hop, per-flow stateless operation, required for large-scale networks
  - Minimal/no jitter, minimal/no receiver playout buffer
- The time seems to be right:
  - Feasible to implement at high speed with FIFO and/or PIFO options
  - When Damper was thought of, it was infeasible for forwarding planes
- gLBF specifically contributes
  - Per-hop Asynchronous and/or synchronuous operation
  - Use of UBS calculus/queung/scheduling
  - Combined benefits of TSN-ATS and TSN-CQF – for DetNets ?!
- Designed for large-scale / wide area network DetNet deployments
  - But potentially equally able to superceed prior mechanisms in smaller scale networks Based on its benefits

References

References

- LBF: original research paper without guaranteed Latency

  T. Eckert, A. Clemm, S. Bryant, "gLBF: Per-Flow Stateless Packet Forwarding with Guaranteed Latency and Near-Synchronous Jitter", 2021 17th International Conference on Network and Service Management (CNSM)

- gLBF original paper

  T. Eckert, A. Clemm, "High-Precision Latency Forwarding over Packet-Programmable Networks", in 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS 2020)

- Springer Journal 2023 extended paper with more validation data

  T. Eckert, A. Clemm, S. Bryant, "High Precision Latency Forwarding for Wide Area Networks Through Intelligent In-Packet Header Processing (gLBF)", Journal of Network and Systems Management, 31, Article number: 34 (2023)