# 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)

IETF117 DetNet WG meeting 07/26/2023, rev 1.0

https://github.com/toerless/detnet/tree/main/slides

# Overview

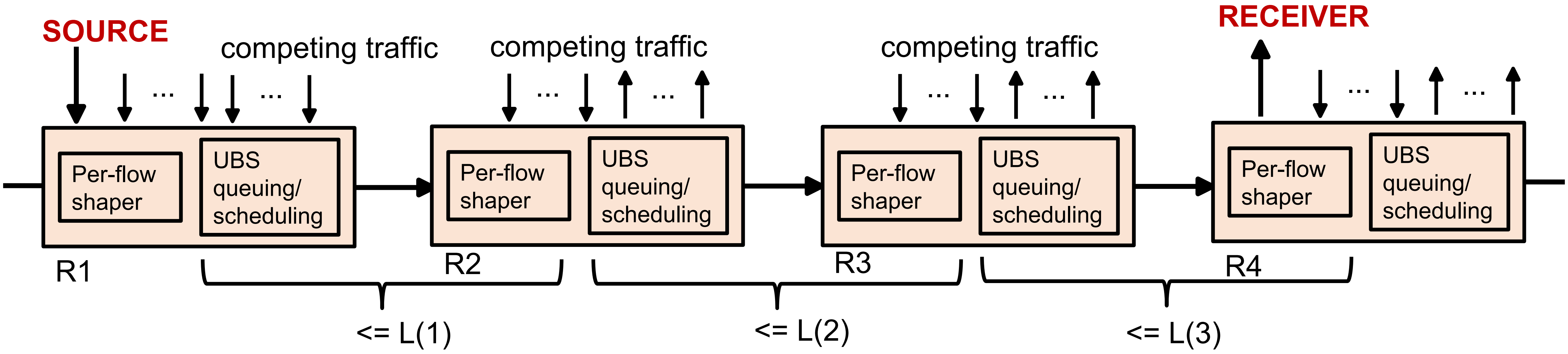
- Detailed step-by-step explanation of gLBF in draft
    - And 07/19/2023 interim
    - See https://wiki.ietf.org/en/group/detnet/wmosq (TBD)
    - https://datatracker.ietf.org/meeting/interim-2023-detnet-08/materials/slides-interim-2023-detnet-08-sessa-deterministic-networking-detnet-data-plane-guaranteed-latency-based-forwarding-glbf-00

- This slide deck focusses on comparison and justification (benefits)
    - Details aspects that where questioned during interim and on WG list.

# What is gLBF

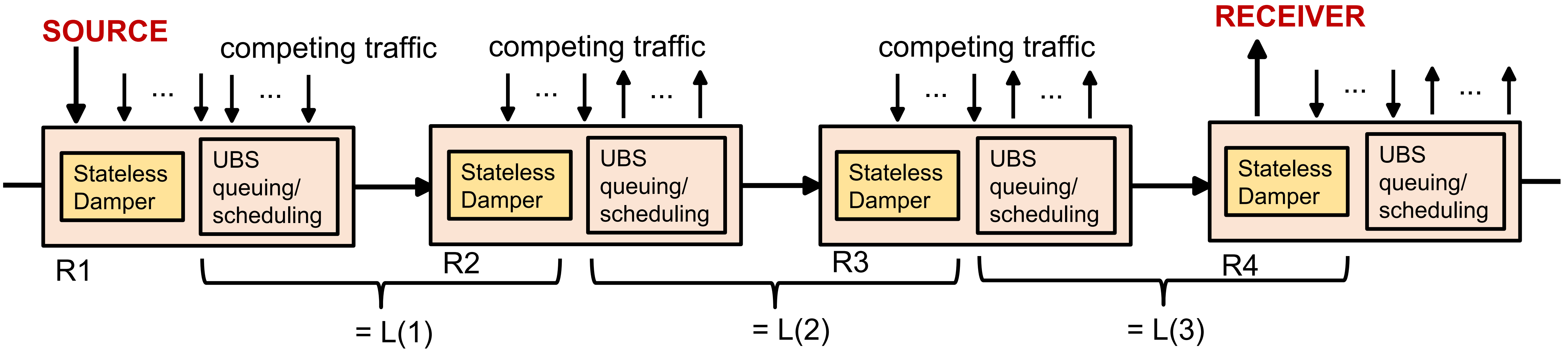
gLBF is UBS (TSB-ATS) with Dampers – instead of shapers (regulators)

# UBS (TSN-ATS)



- Example: R1..R4 in **metro network ring**, each being a spoke to many source/receiver nodes (not shown)
- Relevant end-to-end bounded (guaranteed) queuing latency for Flow F **SOURCE->RECEIVER**:
- L = L(1) + L(2) + (L3)
- Each L(i) calculated from all admitted flows (calculus).
- Without competing traffic, queuing latency for F will be 0.
- With maximum amount of competing traffic, queuing latency for F will be L
- We call this “**in-time**” – packets can arrive as early as possible, but never later than bounded latency.
- Shapers do not increase the per-hop bounded latency. They ensure that the per-hop latency can be calculated independently of the latency across any other hop!

# gLBF

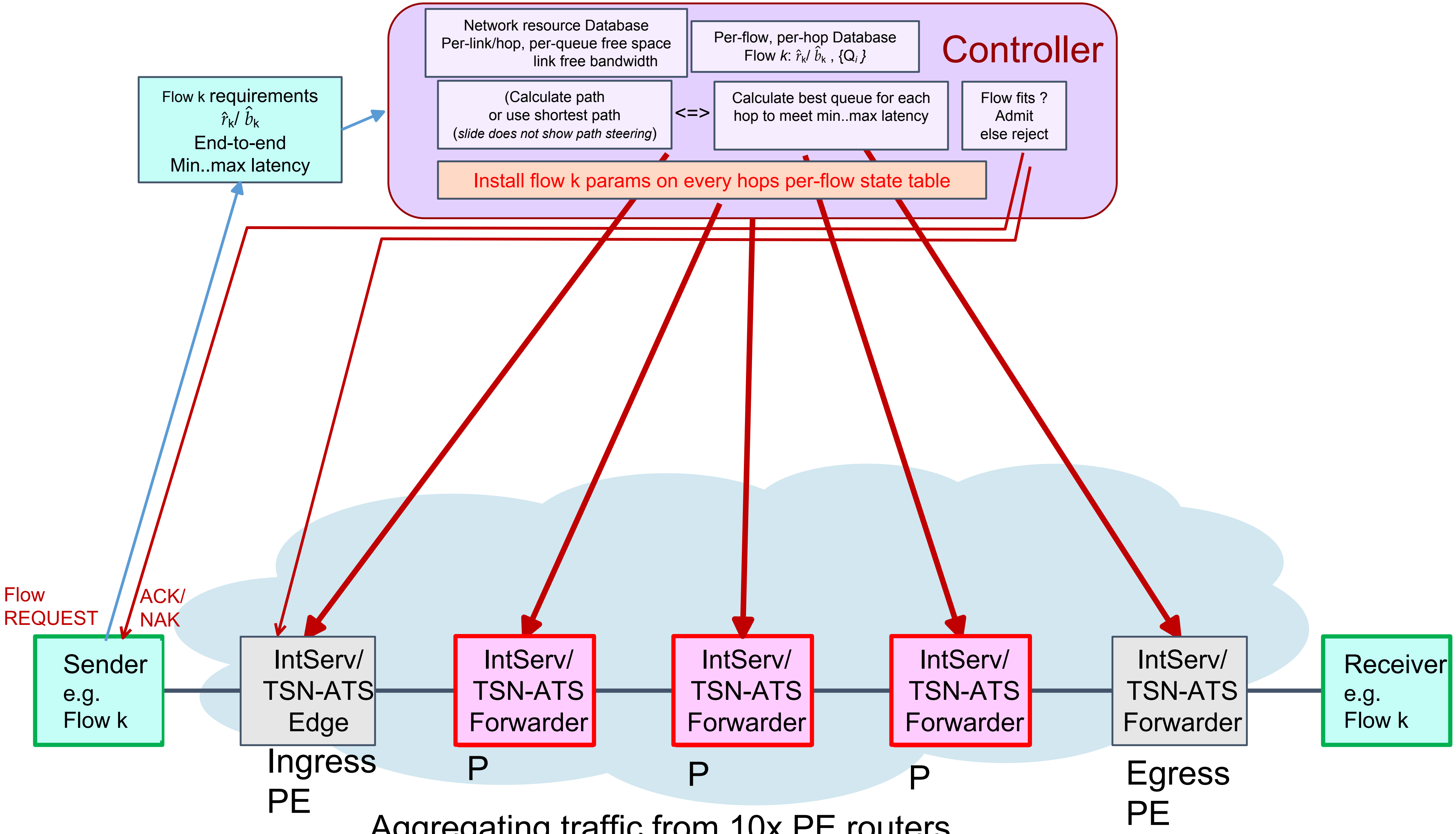


- gLBF replaces shapers with stateless damper
- End-to-end bounded, (guaranteed) queuing latency for Flow F **SOURCE->RECEIVER**:
- L = L(1) + L(2) + (L3)
- Each L(i) calculated from all admitted “competing” flows (UBS calculus) – **EXACTLY AS FOR UBS**.
- Latency through UBS queue/scheduling with be exactly as in UBS – depending on amount of competing traffic
- But latency through UBS queu/scheduler plus following Damper will always exactly be L(i)
- We call this “**on-time**” – packets can arrive as early as possible, but never later than bounded latency.
- Dampers are there to replace shaper and their problems, and they also provide on-time service.

# Why ?

Benefits of Dampers over Shaper/Regulators

# System model with RFC2212/TSN-ATS



Network resource Database
Per-link/hop, per-queue free space
link free bandwidth

Per-flow, per-hop Database
Flow *k*: $\hat{r}_k / \hat{b}_k$ , $\{Q_i\}$

Controller

(Calculate path
or use shortest path
(*slide does not show path steering*)

<=>

Calculate best queue for each
hop to meet min..max latency

Flow fits ?
Admit
else reject

Install flow k params on every hops per-flow state table

Flow k requirements
$\hat{r}_k / \hat{b}_k$
End-to-end
Min..max latency

Flow
REQUEST

ACK/
NAK

Sender
e.g.
Flow k

IntServ/
TSN-ATS
Edge

Ingress
PE

IntServ/
TSN-ATS
Forwarder

P

IntServ/
TSN-ATS
Forwarder

P

IntServ/
TSN-ATS
Forwarder

P

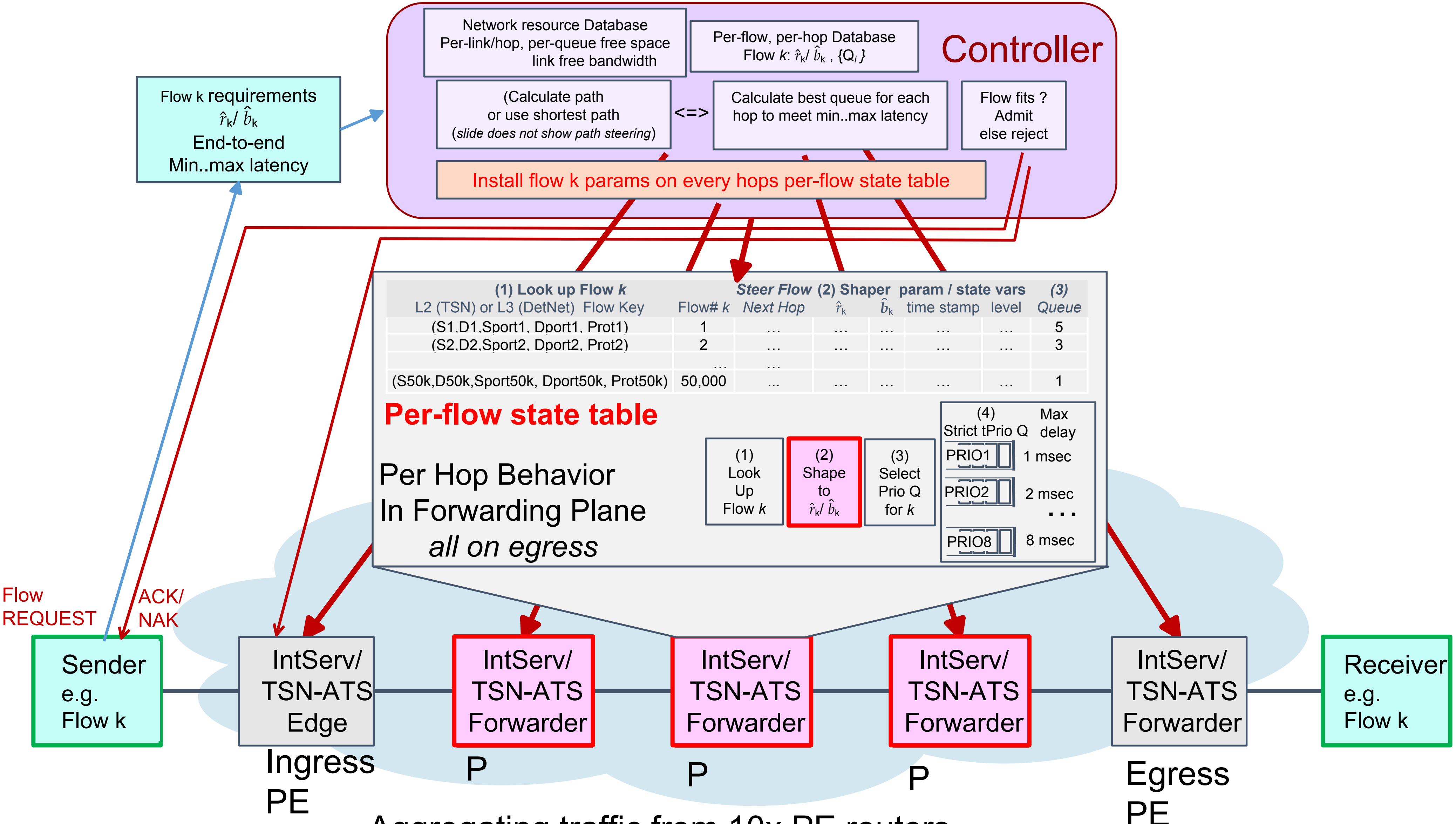
IntServ/
TSN-ATS
Forwarder

Egress
PE

Receiver
e.g.
Flow k

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

# System model with RFC2212/TSN-ATS



Controller

Network resource Database
Per-link/hop, per-queue free space
link free bandwidth

Per-flow, per-hop Database
Flow $k$: $\hat{r}_k$/ $\hat{b}_k$ , {$Q_i$}

(Calculate path
or use shortest path
(*slide does not show path steering*)

<=>

Calculate best queue for each
hop to meet min..max latency

Flow fits ?
Admit
else reject

Install flow k params on every hops per-flow state table

Flow k requirements
$\hat{r}_k$/ $\hat{b}_k$
End-to-end
Min..max latency

| (1) Look up Flow *k* | | *Steer Flow* | (2) Shaper | param / state vars | | | *(3)* |
|---|---|---|---|---|---|---|---|
| L2 (TSN) or L3 (DetNet) Flow Key | Flow# *k* | *Next Hop* | $\hat{r}_k$ | $\hat{b}_k$ | time stamp | level | *Queue* |
| (S1,D1,Sport1, Dport1, Prot1) | 1 | … | … | … | … | … | 5 |
| (S2,D2,Sport2, Dport2, Prot2) | 2 | … | … | … | … | … | 3 |
| | … | … | | | | | |
| (S50k,D50k,Sport50k, Dport50k, Prot50k) | 50,000 | ... | … | … | … | … | 1 |

**Per-flow state table**

Per Hop Behavior
In Forwarding Plane
*all on egress*

(1)
Look
Up
Flow *k*

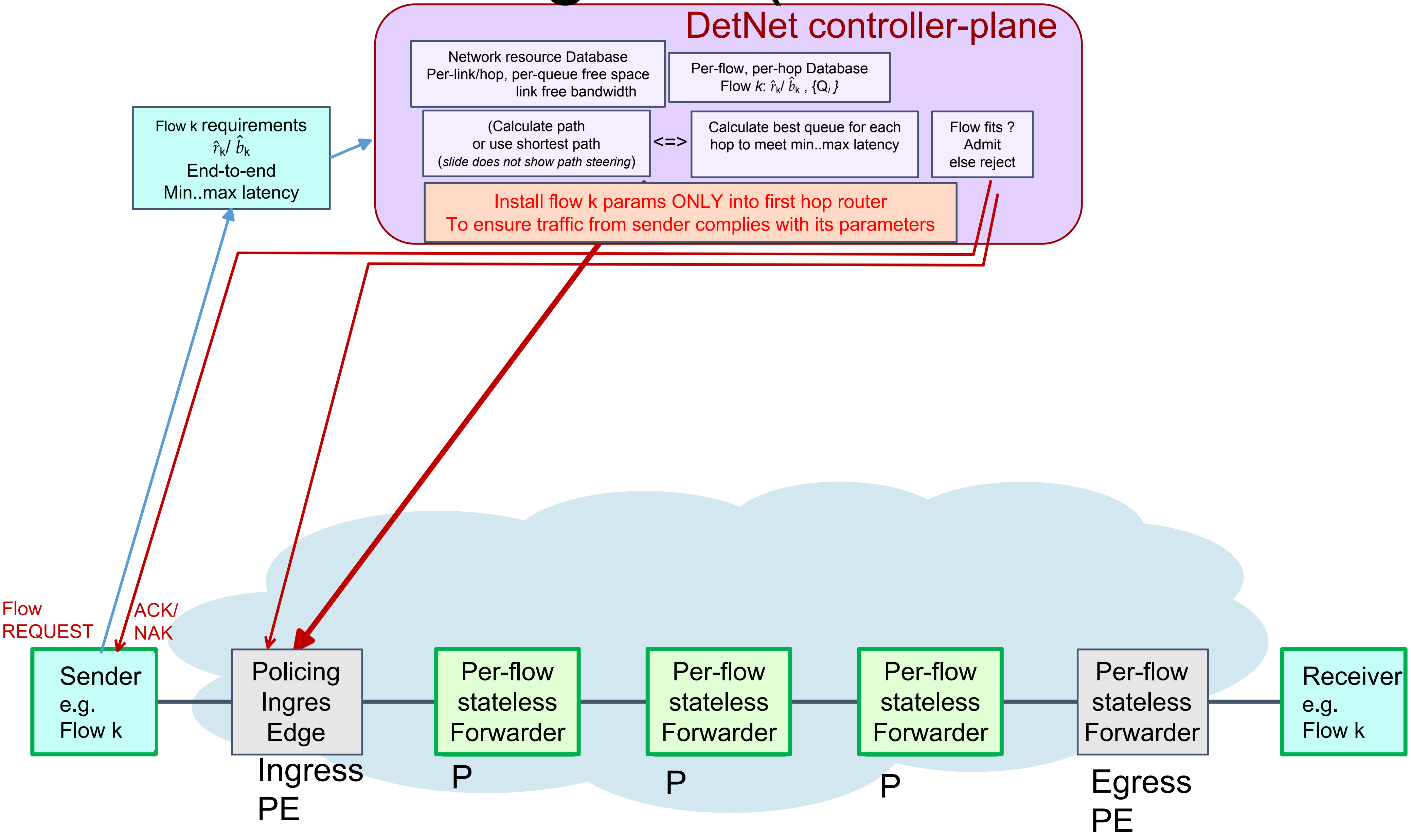
(2)
Shape
to
$\hat{r}_k$/ $\hat{b}_k$

(3)
Select
Prio Q
for *k*

| (4)<br>Strict tPrio Q | Max<br>delay |
|---|---|
| PRIO1 | 1 msec |
| PRIO2 | 2 msec |
| … | … |
| PRIO8 | 8 msec |

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

# System model with gLBF (or TCQF/CQSF/...)



DetNet controller-plane

Network resource Database
Per-link/hop, per-queue free space
link free bandwidth

Per-flow, per-hop Database
Flow *k*: $\hat{r}_k / \hat{b}_k$ , {$Q_i$}

(Calculate path
or use shortest path
(*slide does not show path steering*)

<=>

Calculate best queue for each
hop to meet min..max latency

Flow fits ?
Admit
else reject

Install flow k params ONLY into first hop router
To ensure traffic from sender complies with its parameters

Flow k requirements
$\hat{r}_k / \hat{b}_k$
End-to-end
Min..max latency

Flow
REQUEST

ACK/
NAK

Sender
e.g.
Flow k

Policing
Ingres
Edge

Ingress
PE

Per-flow
stateless
Forwarder

P

Per-flow
stateless
Forwarder

P

Per-flow
stateless
Forwarder

P

Per-flow
stateless
Forwarder

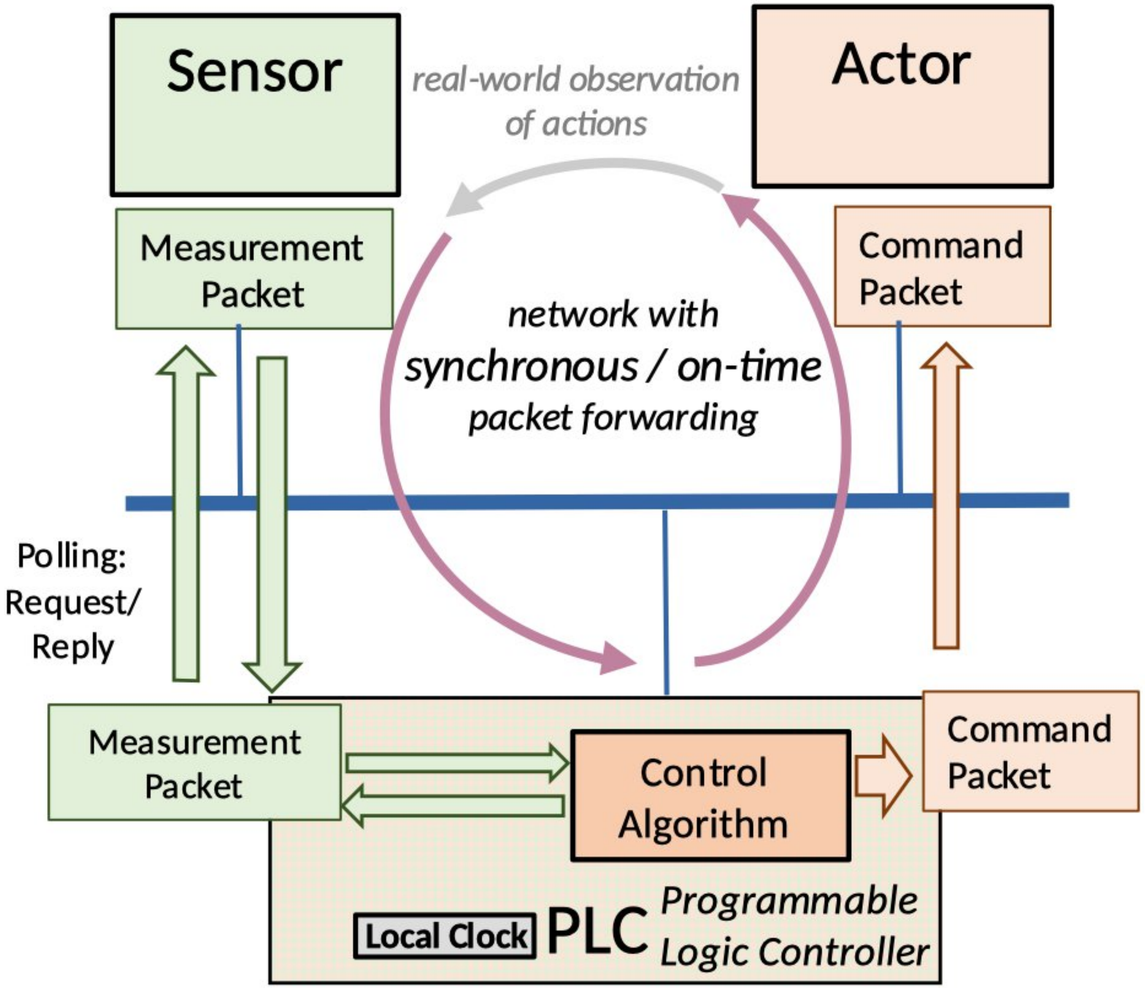
Egress
PE

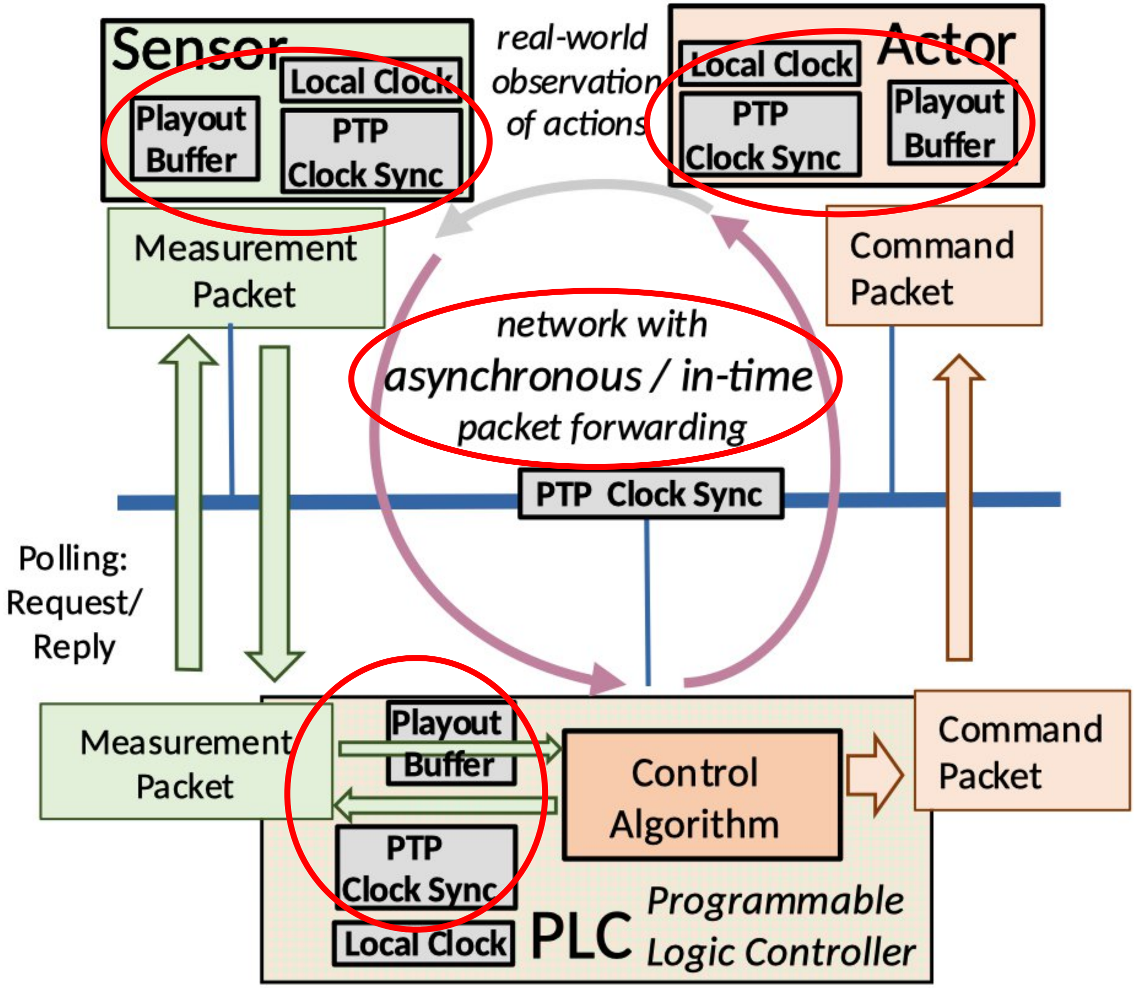
Receiver
e.g.
Flow k

# Why gLBF instead of UBS/RFC2212
# 1. The stateful shaper (regulator) problems

- Operate (download/troubleshoot) per-flow shaper state on every hop.
  - Increases control-plane load N-times (N=#hops), increases troubleshooting complexity
  - SPs replaced RSVP-TE with SR to exactly eliminate these problems. And Shaper state is more complex than RSVP-TE “steering state”.
  - Similarily in multicast, steering/replication state is being eliminated by BIER because of state complexity.
- Cost / feasibility of high-speed and large number of hardware shaper
  - Shaper require read/compute/write cycles to shaper state memory – which may need to have hundred of thousand of entries.

# Benefits of on-time (synchronous)





*Resynchronizing packets via playout buffer and clock synchronization*

(C) 2023, Springer, Journal of Network and Systems Management, 31, Article number: 34 (2023)

# Why gLBF instead of UBS/RFC2212
# 2. On-time is most often better than in-time

- Reduced need for clock synchronization
  - On-time allows lightweight devices to operate without clock synchronization
    – because they do not need playout buffering. This simplifies applications / control-loops
  - Can eliminate need for network to provide “clock synchronization as a service” to applications
- No need for network size dependent playout buffering
  - Network takes care of buffering. Applications do not need to be designed with specific network size (playout buffer) in mind.

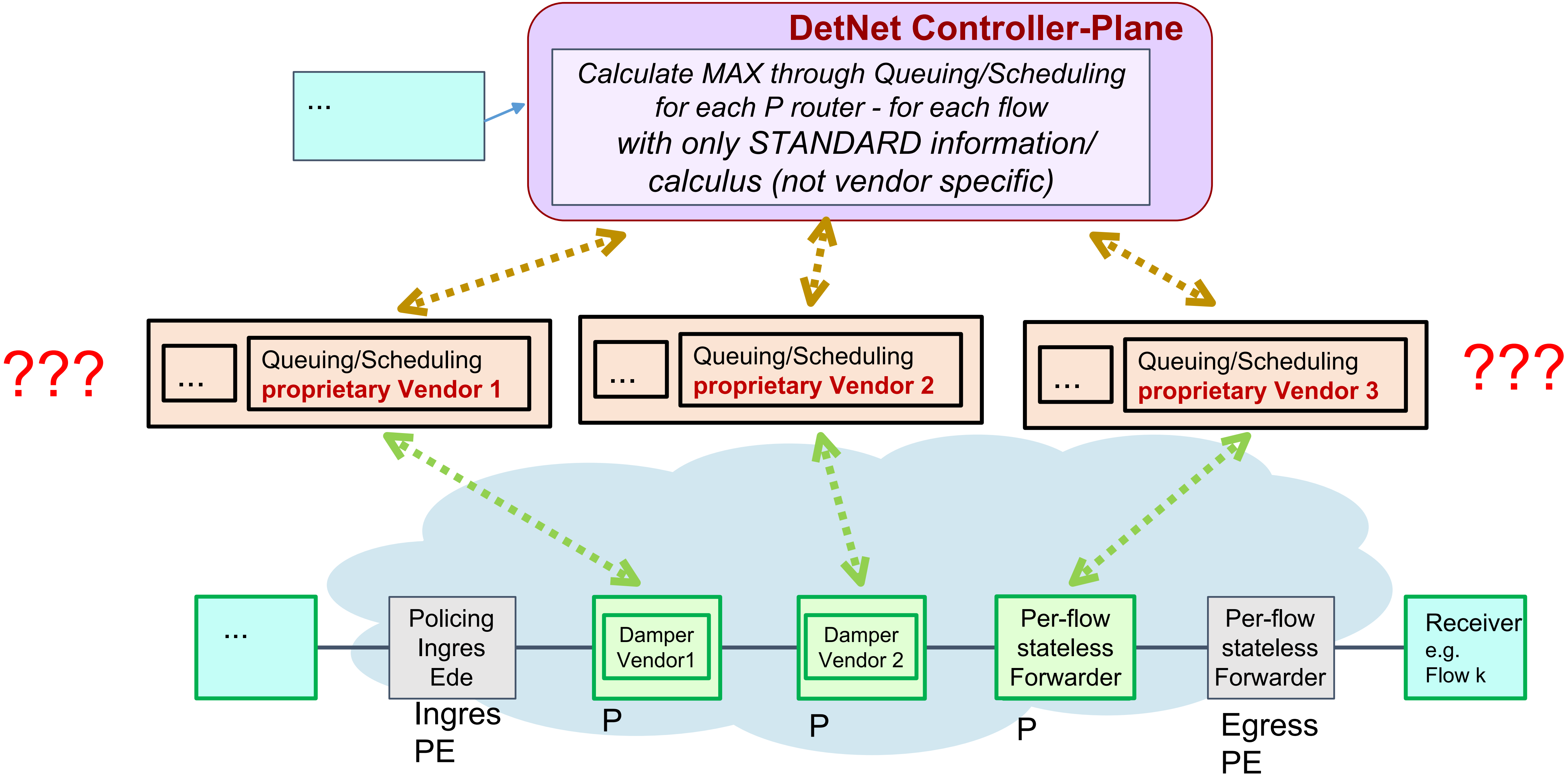
AFAIK: in-time services (guaranteed bouneded latency!) always requires shapers/interleavers (not a UBS specific issue!).

In large networks, this will be most cost/operations expensive.

# Why not Damper with ANY queuing/scheduling ?

Benefits of using UBS

# Standard system requirement


DetNet Controller-Plane
Calculate MAX through Queuing/Scheduling for each P router - for each flow with only STANDARD information/ calculus (not vendor specific)
…
???
…
Queuing/Scheduling proprietary Vendor 1
…
Queuing/Scheduling proprietary Vendor 2
…
Queuing/Scheduling proprietary Vendor 3
???
…
Policing Ingres Ede
Damper Vendor1
Damper Vendor 2
Per-flow stateless Forwarder
Per-flow stateless Forwarder
Receiver e.g. Flow k
Ingres PE
P
P
P
Egress PE

# Why not Damper with ANY queuing/scheduling ?

- We can not build a standardized control plane without picking a queuing/scheduling with standardized, public validated calculus.

- We can not know whether Damper + “Some Queuing/Scheduling” can be implemented at high-speed/low-cost.

- UBS is the most simple calculus (easier than RFC2212) with well vetted calculus
  - But yet extremely flexible to manage latency for each flow.

- Using UBS allows to reuse / share much of controller plane that may be built for TSN-ATS
  - Includes complex network-wide bandwidth+latency path + queue(priority) selection algorithms
  - Which may use backtracking or Deep Learning.

- Should even be of interest to small-scale networks (replacing UBS)
  - With minimal overall design changes.

# Positioning

# Positioning

- Proposed as 2nd gen solution after TCQF/CSQF
    - These methods already provide on-time, stateless forwarding
      And have high-speed, low-cost large-scale network validation.
    - gLBF Adds more flexibility (UBS calculus instead of fixed cycles) and elimination of per-hop clock-synchronization
    - TCQF / CSQF solutions / deployment should be able to easily evolve / migrate into gLBF
        - E.g.: configure gLBF buffers to be same size on each hop: ~= TCQF !

- Why longer term ?
    - gLBF requires new network packet header metadata (delay value)
        - TCQF (TC, DSCP) and CSQF (SRH) work with existing headers! Faster to adopt!
    - gLBF has no high-speed implementation, wide-area deployment validation yet
        - But 2 high-speed algorithm implementation proposals we think can be done at “Low cost”.

# Validation

# Validation

- Simulation with 2 hops (minimum setup) to prove that without Damper (or Shaper), we can not calculate the bounded latency
  - Flows will exceed their calculated latency
- With Damper inserted, flows again will be forwarded within calculated maximum queuing latency (in Router4 FIFO).

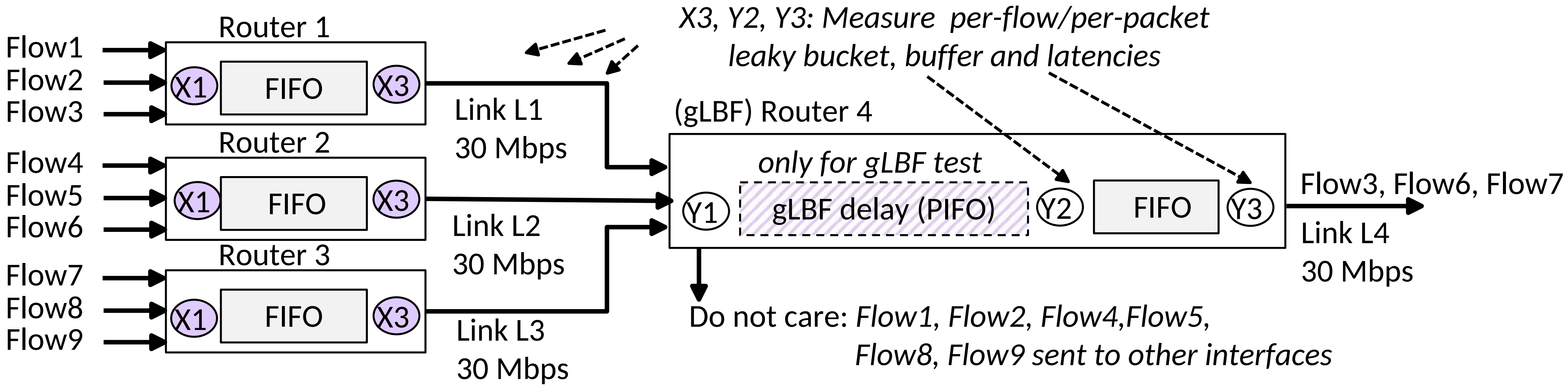


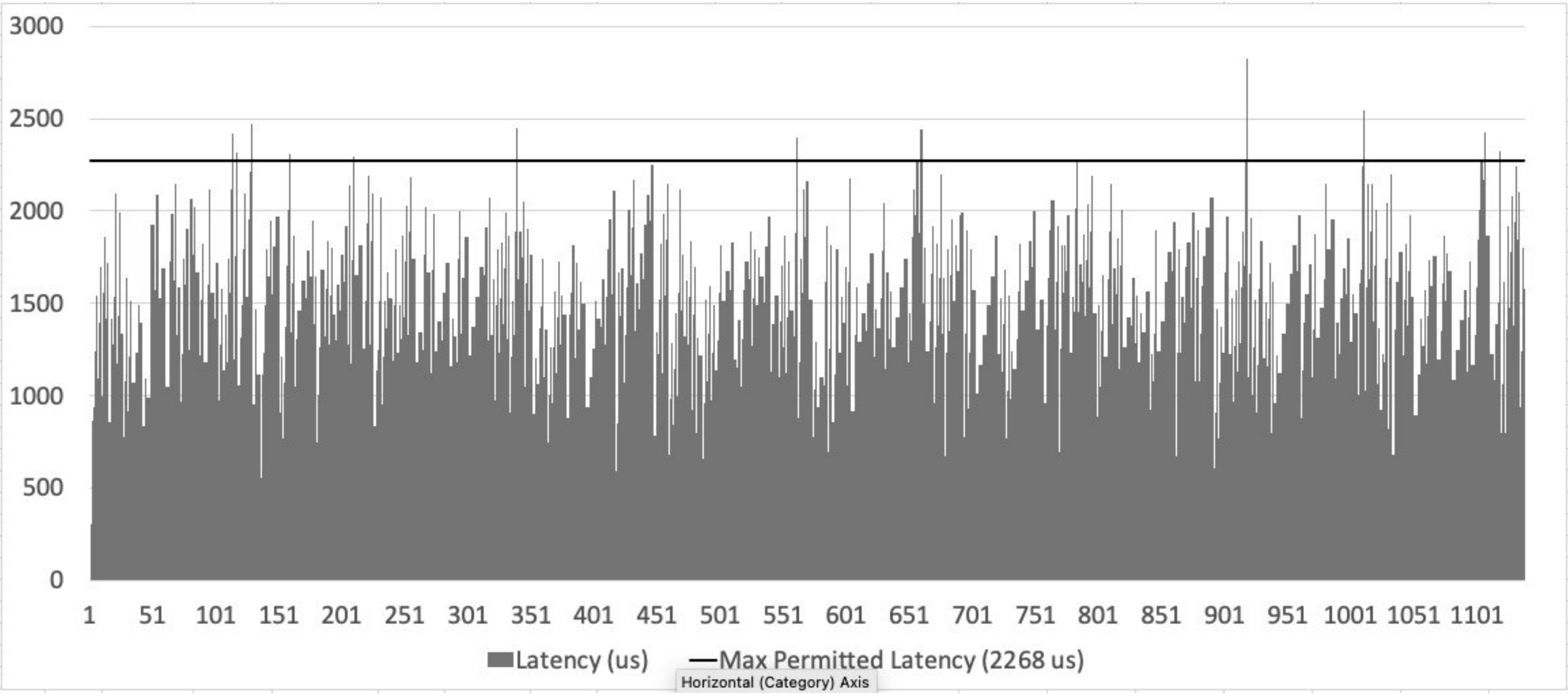
Image (C) from 2020 IEEE/IFIP Network Operations and Management Symposium (NOMS 2020)

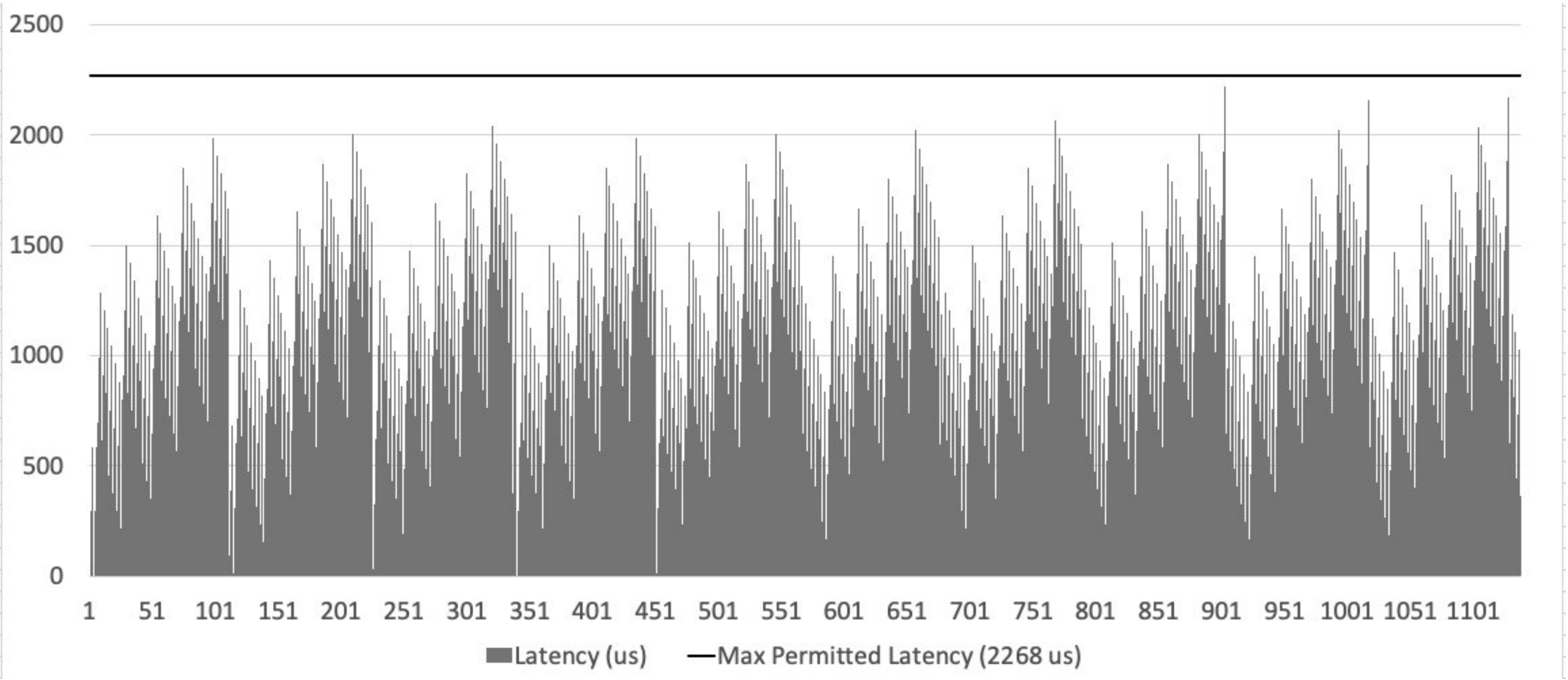
# Validation

Latency of packets of one of the 9 flows across Router 4 FIFO queue.

Upper picture: no Damper, some packets exceed bounded latency

Lower picture: Damper added, all packets stay within calculated limit.





Images (C) 2023, Springer, Journal of Network and Systems Management, 31, Article number: 34 (2023)

# Questions ?

# Backup Slides

# gLBF model: metadata, forwarding, state

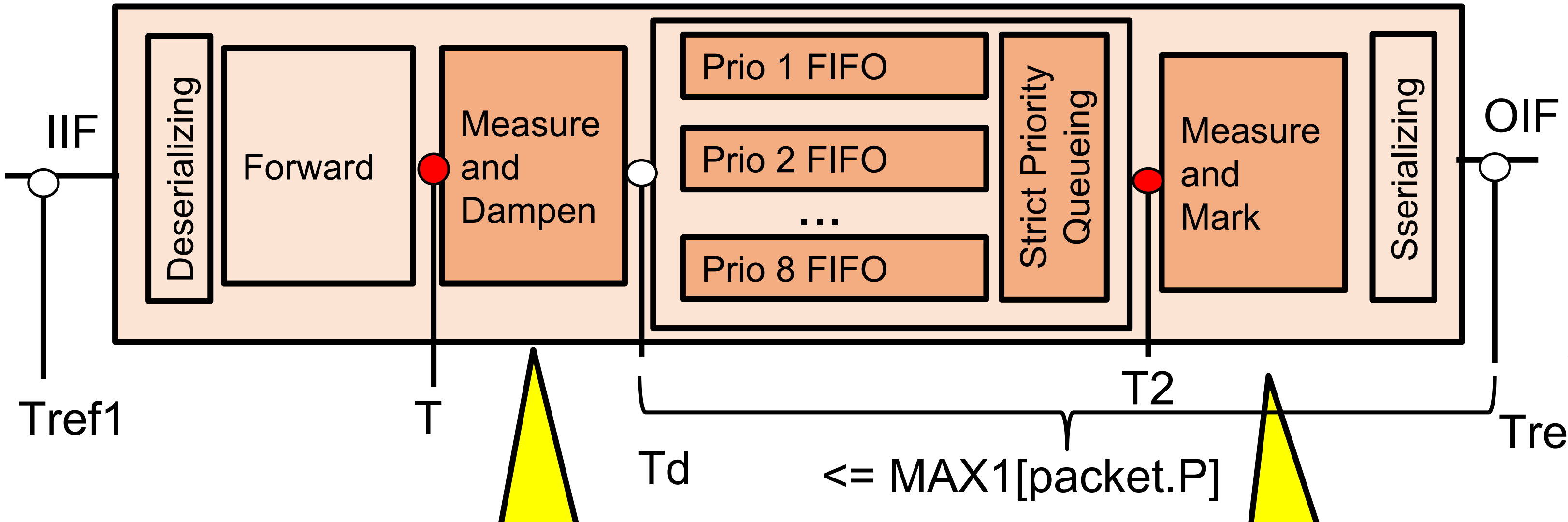
● Measured time

○ Calculated time

Packet Metadata

P(rio):
end-to-end or
per-hop (eg:SIDs)

D(elay) [nsec]



Per-OIF state:

Prio 1: MAX1[1]
Prio 2: MAX1[2]
...
Prio 8: MAX1[8]

Tref1 = T - packet.psize[bits] / IIF.speed [bps]
Td = Tref1 + packet.D

Tref2 = T2 + packet.psize[bits] / OIF.speed [bps]
packet.D = MAX1[packet.P] - (Tref2 - Td)

- 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.