DETNET
Internet-Draft
Intended status: Informational
Expires: 8 January 2026

T. Eckert
Futurewei Technologies USA

7 July 2025

Deterministic Networking (DetNet) Data Plane - Flow interleaving for scaling detnet data planes with minimal end-to-end latency and large number of flows.

draft-eckert-detnet-flow-interleaving-03

Abstract

This memo explain requirements, benefits and feasibility of a new DetNet service function tentatively called "flow interleaving". It proposes to introduce this service function into the DetNet architecture.

Flow interleaving can be understood as a DetNet equivalent of the IEEE TSN timed gates. Its primary role is intended to be at the ingress edge of DetNet domains supporting higher utilization and lower bounded latency for flow aggregation (interleaving of flows into a single flow), as well as higher utilization and lower bounded latency for interleaving occurring in transit hops of the DetNet domain in conjunction with in-time per-hop bounded latency forwarding mechanisms.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 8 January 2026.

Table of Contents

1. Introduction

1.1. Overview and summary

This memo explain requirements and benefits of a new DetNet service function tentatively called "flow interleaving" in this memo. It proposes to introduce this service function into the DetNet architecture.

Flow interleaving can be understood as a DetNet equivalent of the IEEE TSN timed gates. Its primary role is intended to be at the ingress edge of DetNet domains supporting higher utilization and lower bounded latency for flow aggregation (interleaving of flows into a single flow), as well as higher utilization and lower bounded latency for interleaving happening in transit hops of the DetNet domain in conjunction with in-time per-hop bounded latency forwarding mechanisms.

1.1.1. Background

Currently, DetNet has a set of functions/services including Packet Replication, Elimination and Ordering for resilient transmission of DetNet packets over failure disjoint paths. DetNet is currently relying on pre-existing forwarding plane functions from other efforts such as IEEE TSN and prior IEEE work such as [RFC2211] and TBD control-plane functions for guaranteeing deterministic bounded latency with (near) zero loss and bounded latency. DetNet is also as of this writing (mid 2023) in the process of investigating DetNet specifications of additional forwarding plane methods in support of bounded latency and jitter, especially for large scale DetNets.

As in suport of such scaling goals, it is important to not only consider per-hop mechanisms to support scaling, but also ingress node processing. Flow interleaving as this memo call one such function is one such function.

1.1.2. Avoiding burst across multiple hops

The core challenge with bounded latency guarantees is that traffic flows are by design bursty, and the end-to-end latency that any hop-by-hop forwarding mechanism can guarantee (DetNet, TSN, ...) depends on the maximum amount of packets that may collide anywhere in the network on an interface, and cause queuing/scheduling delay of packets.

In typical DetNet topologies, such as metropolitan access rings used for residential/industrial wireline subscribers as well as mobile network towers, this problem can occur worst case on every hop, which could be 20 or more hops. When these bursts are not controlled, a lot of latency can occur unnecessarily. This is the same problem of

no-coordination as the latency inherited at roadway intersections with car traffic: The total amount of traffic on the streets is far from capacity, but the intersecting traffic occurs exactly when ones own car wants to move, resulting in unnecessary delay. In the recent decade there has terefore been a good amount of interest in elliminating those traffic-red-light caused delays through the use of autonomic cars who could be crossing each other at intersections without collisions, such as in https://www.youtube.com/watch?v=r7_lwq3BfkY.

The same non-queuing mechanisms have been used in computer networking for decades via so-called Time-Division-Multiple-Access mechanisms, primarily for bitstream type channels of data. In TSN, this mechanism is achieved through so-called "Gates", that allow to excatly time the periodc windows in time when TSN flows are allowed to send packets into the network / next-hop. Note that TSN gates are a very flexible mechanism used for different purposes.

2. Problem and use-case analysis

2.1. Single hop burst aggregation

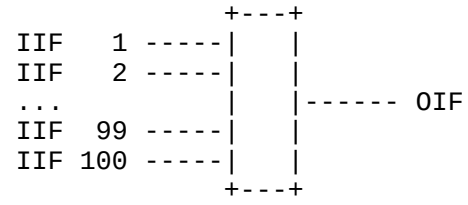

Figure 1: Single Hop burst aggregation

Consider in Figure 1 a network node receiving detnet traffic that requires bounded latency from 100 Incoming InterFaces (IIF) that all has to exit on one Outgoing InterFace (OIF).

When each of these IIF has a packet at the same time destined to OIF, then these packets have to be queued up before they can be sent out on OIF. Assuming IIF and OIF are all the same speed and the packets all of the same size, then the worst queuing latency for a single packet introduced is 100 times the serialization time of a single packet.

Assume each of the IIF carries 10 flows, each of which wants to send one 1500 byte sized packet once every 20 msec. Such a frequency of messages would for example happen in video based control loops, where a reaction happens once every video fraem delivered, which could be 20 msec for the frame rate of 50 Hz. In reality, higher frame rates such as 60/90/120 are more common these days, but 50 is easier to use in examples.

If all these flows send their packets uncoordinated or coordinated simultaneously, then the worst case is that each of the 100 interfaces has 100 back-to-back bursts of 10 packets. In this case, the worst-case queuing latency is 12 msec for a packet.

This queuing of course is undesirable because the total required rate for all these 100 IIF * 10 Flows/IIF = 1,000 total flows is just 600 Mbps,so there is no bandwidth congestion. If all these flows packets would arrive nicely interleaved, none of them would experience any queuing latency. Assume [TCQF] was used with a cycle time of 20 usec, and each of the 1000 flows packets would be put into a separate cycle: 1,000 cycles of 20 usec each fit exactly the 20 usec period, so each flows packet would experience just a 20 usec queuing latency instead of potentially 12 msec.

2.2. Ingress interleaving

Consider the router in Figure 1 is the ingress router into a DetNet domain and each IIF is connected to some IoT device such as a sensor, that is periodically sending a sensor data packet. Assume all these traffic flows would even need to go to a single receiver, such as a PLC or environmental control system. This ends up being a situation such as shown in Figure 2.

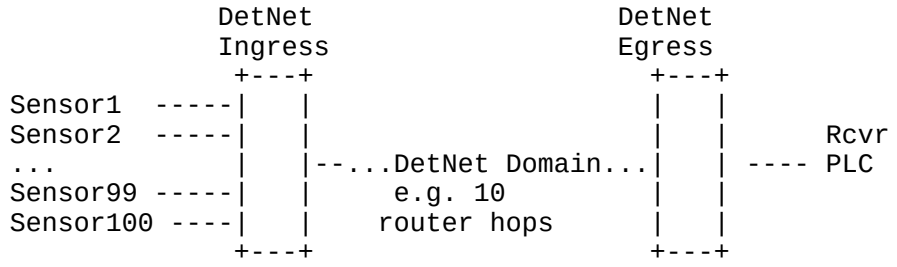

Figure 2: Ingress aggregation use case example

Whether or not the flows are sent into the DetNet in some aggregated fashion or not:

If the packets of these flows packets arrive uncoordinated at the DetNet Ingress router, the maximum burst size of an individual flow, and this burst size is not only relevant for the maximum latency through this ingress router, but also for the maximum latency that these 100 flows may at wors incur on other hops along the path (as described in more detail in the following sections).

If instead the packets of these 100 flows are interleaved "nicely" such that the packets of all flows are sent at a different offset into for example a common period time (such as 20msec), then the maximum burst size that any DetNet would have to account for would be 1/100 times as large. End-to-end latency that could be guaranteed would be lower and utlization higher.

Such "nice" interleaving could be done at the application side, such as the PLC triggering the sensors to send sensor data at specific times, or programming them to send that periodically with those different offsets to avoid packets arriving at the same time.

In a large DetNet, or simply a small DetNet that is not fully trying applications to perform such functions, this approach is not feasible. In a large DetNet for example there may be no relevant single PLC that could coordinate sending times, but instead a large number of independent applications would multiplex without any common coordination.

Instead, the DetNet ingress router would have to perform the interleaving function in the forwarding plane, receiving uncoordinated packets from each flow/sender and making them wait until their time in a cycle arrives, before allowing them to be further processes such as by the common, ingress independent egress DetNet per-hop processing. This waiting is what in TSN is called a gate function.

Of course, the gate function itself will also add latency to packet arriving uncoordinated shortly after the gate for the flow closed. But in all cases, this latency occurs only once along the path and it is always lowe than the period time of the flow. Without such flow interleaving, the total queuing latency caused by uncoordinated bursts could exceed such a cycle time (as described in later sections).

2.3. Flow aggregation

When DetNet per-hop bounded-latency operates hop-by-hop on a per-flow basis such as in [TSN-ATS], scalability can be helped by treating the 100 flows of Figure 2 wit the same ingress and egress router as a single aggregated DetNet flow - whether the packets of the 100 flows aggregated are or are not coordinated. When the packets are coordinated/interleaved, then this flows burst size would be 100 times smaller than if they where uncoordinated - reflecting the latency considerations outlined above at the level of a DetNet flow.o

It is useful to consider one use-case of flow interleaving as a sub-function of the DetNet aggregation function, and this is exactly one goal of this memo. In this use-case, flow interleaving can benefit latency under scalability independent of whichever per-hop DetNet bounded latency forwarding mechanism is used.

2.4. Multi-hop queueing

Going back to Figure 1 and now considering a larger topology, such as in a metropolitan area. A ring of 100 routers R1...R100 each has 100 interface IIF1...IIF100. Each of those interfaces could connect to 100 Edge Router (ERxxyy) each serving 100 subscribers. Such a ring would then connect 1,000,000 subscribers.

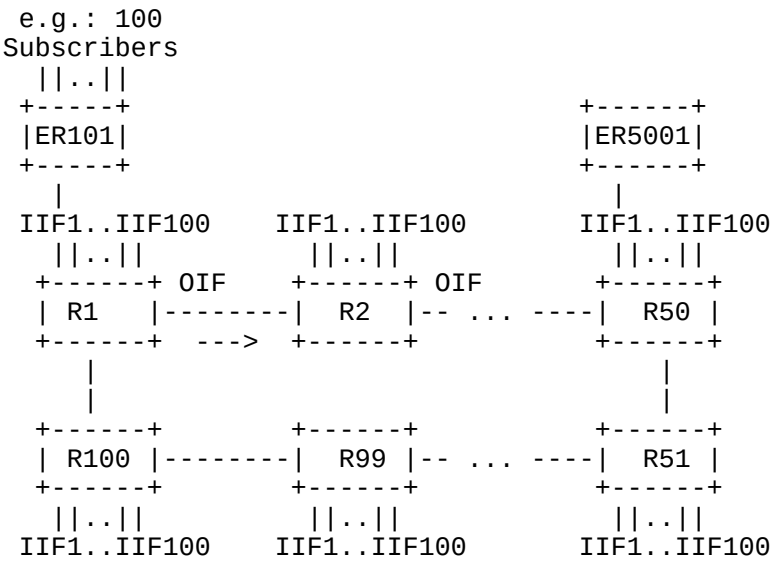

Figure 3: Multi-hop queuing topology

Assume the packet in question is now inserted from ER101 via IIF1 into R1 and travels the ring clockwise to R50 where it exits the ring towards ER5001. On each of the 59 OIF interfaces in the ring it could worst case experience the same worst case bounded delay in the order of what we calculated for the single router setup. For example a large number of such competing traffic flows could go from an ER connected to R1 to an ER connected to R2. Those flows would create the queue on OIF of R1.

Likewise there could be similar flows from R2 to R3, from R3 to R4 and so on. The sum of worst case queue buildups is thus proportional to the number of hops traversed. And of course nobody is interest in a bounded lateny of 49 * 12 = 588 msec, aka: more than half a second. Within a metropolitan area where the non-queuing network latency does not even add up to 1 msec.

While the extreme case is not very likely, this type of aggregation of queuing latency in worst cases woulk in principle not be untypical in target use-cases. If ride-share cars (Uber, DiDi,...) become remote controlled and subscribers to the networks are either such remote drivers from home or radio towers connecting to the remote controlled cars, thousands, if not tenth of thousands of such flows may co-exist. And one certainly does not want driving to become slower and slower the further away from the driver the car is - not because of speed of light, but because of unnecessary queuing, higher RTT and hence lower speed for the car that still alows the driver to react fast enough to avoid accidents.

2.5. Multi-hop flow interleaving

In the same way as in the single-hop flow interleaving on ingress, packets of flows can also be interleaved on ingress but now considering not only that they do not collide on the outgoing interface of this ingress router, but also considering them competing at the same time with packets arriving from other interfaces on some hops further down the path. This sounds more complex than it can be in practice, as explained later in the document.

2.6. Note: Multi-hop burst accumulation

The problems described in the prior section is not to be mixed up with "multi-hop burst accumulation" as explained here. Burst accumulation refers to the fact that because of the aforementioned queuing delay due to simultaneously occurring traffic, the burstyness of individual flows can increase. And this then can lead to further problems downstream.

Consider the prior section network setup and the same flow which sends a packet once every 20 msec. Assume that packet n of this flow experiences on two consecutive hops a queuing latency of 10 msec each because of competing traffic. But now this competing traffic is intermittent and packet n+1 of the flow passes the same two hops without any queuing delay. Now both the n and n+1 packets of the flow are back to back. And hence the burst size of the flow has doubled. This may cause on downstream hops more delay for other flows than anticipated by admission control, and hence not only invalidating other flows latency guarantees, but on highly loaded links potentially also leading to discarding packets because buffers are overrun.

This burst accumulation is compensated for in bounded latency mechanisms such as [UBS] ([TSN-ATS]) by per-flow shaper/interleaved regulators. In this example case, the shaper would cause the n+1 packet to be delayed by 20 msec because of the late arriving packet n.

In conclusion, compensation of burst accumulation does not eliminate the problem of queue latency accumulation over multiple hops when in-time queuing mechanisms are used and flows are bursty.

2.7. Differences to TSN

In TSN and small-scale DetNet networks, interleaving may be inserted through additional gates (interleave functions) for individual flows on every hop of a path. In large scale DetNet networks, this is highly undesirable due to the target PE/P distinction of path functions. Ideally, per-flow operations including signalling between controller-plane and node as well as advanced traffic-plane functions such as gates should only happen once, on the ingress node to the detnet domain.

2.8. Summary

Flow interleaving is necessary to reduce the per-hop queuing latency and to increase utilization of networks with Deterministic network traffic at lower end-to-end queuing latency.

Interleaving can achieve improvements based on the total number of hops (and hence queues), and depending on how bursty the traffic is. Traffic flows which send packets with a long period of inactivity are the worst case: because of the long period between packets, the network can only support a large number of these flows when these bursts do not occur at the same time but are coordinated so as to cause minimum per-hop latency.

3. Flow interleaving with different per-hop forwarding mechanisms

Building a controller plane to support flow interleaving likely has many possible variations. This section outlines one approach that this memo thinks is simple and scaleable to large number of flows and high rates of flow changes.

4. Overall solution proposal outline

4.1. Principles

Flow interleaving on ingress solely to decorrelate arrival times of packets from different flows on the output interface of the same router seems easy enough, as its consideration and setup is limited to the single router. This use-case of of flow-interleaving can actually be the first stage of scaling up DetNet deployment with minimal complexity. It is also working across all per-hop forwarding mechanisms for bounded latency, in-time and on-time.

Flow interleaving with the goal to decorrelate the arrival time of different flows packets on output interfaces further along the path sounds very complex, but it actually can be quite simple and possible to support in linear time in the controller-plane when taking three considerations into account.

1. The per-hop forwarding along the path is per-hop on-time, so that the time at which packets arrive on every hop can be calculated accurately by the controller-plane. Such per-hop one-time forwarding methods include [CQF] (if for exampled used with DetNet over TSN solutions), [TCQF], {CSQF}} and [gLBF].

2. The periodicity of traffic flows is some order of magnitude larger than the achievable accuracy of per-hop on-time forwarding. With the examples presented, the periodicity of traffic flows is in the order of msec..100msec, and the accuracy of per-hop on-time delivery in for example [TCQF] can be configured in the order of 10usec or 20usec. In result, the order of granuarity of timing is about 100 times finer than that of application traffic periodicity allowing to decorrelate traffic by up to a factor of 1000.

3. flow interleaving is only an optional optimization mechanism to allow scaling up the use of DetNet traffic in a network which may predominantly carry non-detnet traffic. Allowing to gain another factor of 10 times more DetNet traffic from eg; 1% of nbetwork bandwidth to 10% network bandwidth may be all that is required. One may compare the relatively simple efforts of a controller plane to support flow interleaving with the NP-complete efforts

to optimize useable cpacity of the network for best-effort traffic by NP complete path optimizations - as done in today almost every mayor SP backbone network.

4.1.1. Common assumptions

Assume all traffic flows subject to flow interleacving are described by a burst size of b bits and a period of p [msec]. The b bits are sent back to back as a single packet or burst of packets. p is not allowed to be arbitrary but must be a complete divisor of 100 msec. This allows for traffic flow periods of 1/50, 1/60, 1/90, 1/120 seconds.

Assume (for simplicity) also, that the path for a new flow is calculated without taking flow interleaving into account. For example path selection could use CSPF (Constrained Shortest Path First) or better some optimized selection of a path where the link with the highest utilization has the lowest utilization amongst all alternative paths and the total physcial path propagation latency does not cause the maximum desired latency for the flow to be exceeded.

4.2. Flow interleaving with TCQF

Assume TCQF is being used with 20 usec cycle times. The controller-plane maintain for every outgoing interface in the topology that can be used for TCQF traffic a window of 5,000 cycles and their amount of available bits, not utilized by already admitted flows. The amount of total vits for each cycle depends on the speed of the link and what percentage of the link is allowed to be used by TCQF.

Admitting the flow with interleaving across the previously choosen path consists of finding i = 100msec/p equidistant cycles thus that the choosen cycle with the lowest amount of available bits across the i choosen cycles has the highest number of available bits across all choices for those i cycles. The index for the 5000 cycles of each hop does of course need to taken with a an offset modulo 5000 that reflect the cycle mapping along the path.

```
      // Determine minimum cycle capacity across path
      // Without taking utilization into account

      minc = max_cycle_capcity
      for if in path_hops
        minc = min(cycle_capacity[if], minc)
      minfree[1...5000] = minc

      // Determine for each of the 5000 cycles the [2]
      // minimum free capacity along the path
      ofst, ifp = 0
      for if in path_hops
        ofst += cycle_offset[ifp][if] if ifp
        ifp = if
        for i in 1...5000
          minfree[i] = min(freec[if][((i+ofst) mod 5000)+1],minfree[i])

      // Determine the cycle option 1...nc with the [3]
      // highest free capacity
      bestfree = 0
      bestfreec = -1
      nc = 100msec/p
      d = 5000/nc
      for i in 1...d
        ii = rnd_seq(i,d)
        betterfree = 0
        for j in 0...(nc-1)
          k = (ii + j * d) mod 5000 + 1
          if minfree[k] <= bestfree
            betterfree = 0
            break
          else
            betterfree = min(minfree[k], betterfree)
        if betterfree
            bestfree = betterfree
            bestfreec = ii
```

Figure 4: Flow interleaving controller plane algorithm for TCQF

Figure 4 Shows an example brute-force pseudocode for finding a best set of cycles according to the described conditions.

minfree[1...5000] is initialized in [2] to be the lowest free capacity (e.g.: in bits) for each of the 5000 cycles along the path of the flow. cycle_offset[ifp][if] is the numerical offset o that needs to be applied when a packet arrives with cycle i from interface ifp and is sent out on ifp. It then needs to use cycle ((i + o) % 5000 + 1).

[3] then determines, which of the first d cycles is the best. nc is the number of cycles that the flows burst will fit into the 5000 cycles. rnd_seq randomizes the cycle number so the allocation will not allocate sequential cycles but spread out the flow bursts over time.

In summary, the algorithm is a simple search for which of the set of cycles along the path has the lowest utilization. As a two step proces this is first a linear operation to find the worst case hop for every cycle, an O(pathlength) operation, followed by searching for the best cycle-set, which is O(const), where const is e.g.: 100.

4.3. CSQF

[CSQF] proposes to carry the cycle mapping in the packet header whereas [TCQF] as presented in this memo defines an in-router cycle mapping. Carrying the cycle-mapping in the packet is attractive where it comes for free (or almost free), and this is in segment routing (SR) networks, where packet steering uses already a packet header. Each steering hop is a so-called SID, and if n cycles are to be used, then instead of one SID, n SIDs are allocated for each hop. Especially in IPv6 with 128 bit SIDs, there is no problem to even support large number of cycles.

Flow interleaving across multiple hops with TCQF can easily hit limits in maximum utilization. Consider for example a network where flows with periods of 1/50th and 1/60th seconds occur. It is clear that it will not be possible to achieve equal utilization across all cycles because of the moire effect between these two type of flows.

With CSQF and for example some A more cycles (e.g.: A=4), the controller-plane has the option to choose for every hop of a flow one out of 4 cycles in which the flow would be forwarded. And this number increases exponentially with path length. Hence, the controller-plane ha a lot more opportunity to optimize utilization across cycles - and avoid admission failures at low utilization rates.

4.4. gLBF

glBF with a single priority end-to-end can be used with the same algorithm as shown for [TCQF]. Instead of a fixed cycle-time on every hop, the per-hop latency is independent and depends purely on the admitted maximum burst aggregte across a hop. It is thus more flexible, but utilizing that flexibility would incur more complex controller-plane algorithms. Nevertheless, gLBF could be configured via the controller plane to have the same per-hop latency and therefore allow to be fully backward compatible to [TCQF] with both

latency, jitter and controller-plane algorithm.

Because gLBF itself does not have the notion of cycles, these cycles are on a hop-by-hop basis a result of the timing of packets released by gates on the first-hop routers and packets then delayed by gLBF accurately by the priorities latency on a per-hop basis.

gLBF with multiple priorities and per-hop choice of priority (via appropriate packet headers such as SIDs as used in [CSQF]) would allow to set up similarily if not more flexible flow interleaving as with [CSQF]. - whereas

5. Summary of proposed architectural components

5.1. Forwarding plane gates / "flow interleaver"

Gates derived from TSN gates, adopted to DetNet. Adoption primarily means that these gates would operate logically on ingress, so that they can preceed any pre-existing DetNet per-hop processing, such as that of [RFC2211], [TSN-ATS], [TCQF], {CSQF}}, [gLBF] or any other applicable DetNet per-hop bounded latency mechanism.

Gates also need to preceed an optional DetNet aggregation function in the forwarding plane.

Forwarding plane gates in large-scale DetNets only need to be support on ingress DetNet nodes. In smaller DetNets they may be supported on every hop.

5.2. Controller plane interleaving functions

"Ingress interleave" Controller-plane algorithms to interlave flows in ingress purely for avoiding bursts on the outgoing interface of the same ingress router. result of the algorithm is a set up of ingress gates. These algorithms benefit / can be used with any per-hop forwarding mechanism.

"Fixed network wide interleave" Controller-plane algorithms to interleave all flows in the network and delaying packets solely fvia ingress gates / flow interleaving. This is applicable to all per-hop on-time forwarding methods in the detnet that allow to calculate the fixed time interval at which a packet will be present on every hop of its path.

"Variable network wide interleave" Controller plane algorithm such as above discussed for [CSQF] or [gLBF] which not only calculate a gate parameter for the ingress router, but also parameters for the header of the per-hop mechanism influencing the per-hop latency forwarding of the packets of a flow.

5.3. Controller plane application integration

With flow interleaving resulting in additional first-hop latency which may be significant, it is likely highly beneficial to design appropriate service interfaces between applications that require multiple different flows and the controller-plane to understand the application desired relationship between the phases of packets from different flows of the same application.

(TBD example).

6. Changelog

[RFC-editor: please remove this section]

02 refresh/no textual changes - waiting for progress in design discussion

01 refresh/no textual changes - waiting for progress in design discussion

00 initial version

7. References

7.1. Normative References

[RFC2211] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", RFC 2211, DOI 10.17487/RFC2211, September 1997, <https://www.rfc-editor.org/rfc/rfc2211>.

7.2. Informative References

[CQF] IEEE Time-Sensitive Networking (TSN) Task Group., "IEEE Std 802.1Qch-2017: IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 29: Cyclic Queuing and Forwarding (CQF)", 2017.

[CSQF] Chen, M., Geng, X., Li, Z., Joung, J., and J. Ryoo, "Segment Routing (SR) Based Bounded Latency", Work in Progress, Internet-Draft, draft-chen-detnet-sr-based-

bounded-latency-03, 7 July 2023, <https://datatracker.ietf.org/doc/html/draft-chen-detnet-sr-based-bounded-latency-03>.

[gLBF] Eckert, T. T., Clemm, A., Bryant, S., and S. Hommes, "Deterministic Networking (DetNet) Data Plane - guaranteed Latency Based Forwarding (gLBF) for bounded latency with low jitter and asynchronous forwarding in Deterministic Networks", Work in Progress, Internet-Draft, draft-eckert-detnet-glbf-04, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-eckert-detnet-glbf-04>.

[IEEE802.1Q] IEEE 802.1 Working Group, "IEEE Standard for Local and Metropolitan Area Network — Bridges and Bridged Networks (IEEE Std 802.1Q)", doi 10.1109/ieeestd.2018.8403927, 2018, <https://doi.org/10.1109/ieeestd.2018.8403927>.

[IEEE802.1Qbv] IEEE Time-Sensitive Networking (TSN) Task Group., "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic (TAS)", 2015.

[TCQF] Eckert, T. T., Li, Y., Bryant, S., Malis, A. G., Ryoo, J., Liu, P., Li, G., Ren, S., and F. Yang, "Deterministic Networking (DetNet) Data Plane - Tagged Cyclic Queuing and Forwarding (TCQF) for bounded latency with low jitter in large scale DetNets", Work in Progress, Internet-Draft, draft-eckert-detnet-tcqf-07, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-eckert-detnet-tcqf-07>.

[TSN-ATS] Specht, J., "P802.1Qcr - Bridges and Bridged Networks Amendment: Asynchronous Traffic Shaping", IEEE , 9 July 2020, <https://1.ieee802.org/tsn/802-1qcr/>.

[UBS] Specht, J. and S. Samii, "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks", IEEE 28th Euromicro Conference on Real-Time Systems (ECRTS), 2016.

Author’s Address

Toerless Eckert
Futurewei Technologies USA
2220 Central Expressway
Santa Clara, CA 95050
United States of America
Email: tte@cs.fau.de