

Workgroup: DETNET

Internet-Draft:

draft-eckert-detnet-mpls-tc-tcwf-00

Published: 8 September 2021

Intended Status: Standards Track

Expires: 12 March 2022

Authors: T. Eckert

S. Bryant

Futurewei Technologies USA    University of Surrey ICS

**Deterministic Networking (DetNet) Data Plane - MPLS TC Tagging for  
Cyclic Queuing and Forwarding (MPLS-TC TCWF)**

**Abstract**

This memo defines the use of the MPLS TC field of MPLS Label Stack Entries (LSE) to support cycle tagging of packets for Multiple Buffer Cyclic Queuing and Forwarding (TCWF). TCWF is a mechanism to support bounded latency forwarding in DetNet network.

Target benefits of TCWF include low end-to-end jitter, ease of high-speed hardware implementation, optional ability to support large number of flow in large networks via DiffServ style aggregation by applying TCWF to the DetNet aggregate instead of each DetNet flow individually, and support of wide-area DetNet networks with arbitrary link latencies and latency variations.

**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 12 March 2022.

**Copyright Notice**

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

- [1. Introduction](#)
- [2. Using TCQF with DetNet MPLS \(informative\)](#)
- [3. Data model and tag processing for MPLS TC TCQF \(normative\)](#)
- [4. TCQF with labels stack operations \(normative\)](#)
- [5. TCQF Pseudocode \(normative\)](#)
- [6. TCQF YANG Model \(normative\) TBD](#)
- [7. Computing cycle mappings \(informative\)](#)
- [8. Security Considerations](#)
- [9. IANA Considerations](#)
- [10. Informative References](#)
- [Authors' Addresses](#)

## 1. Introduction

Cyclic Queuing and Forwarding [CQF], is an IEEE standardized queuing mechanism in support of deterministic bounded latency. See also [[I-D.ietf-detnet-bounded-latency](#)], Section 6.6.

CQF benefits for Deterministic QoS include the tightly bounded jitter it provides as well as the per-flow stateless operation, minimizing the complexity of high-speed hardware implementations and allowing to support on transit hops arbitrary number of DetNet flow in the forwarding plane because of the absence of per-hop, per-flow QoS processing. In the terms of the IETF QoS architecture, CQF can be called DiffServ QoS technology, operating only on a traffic aggregate.

CQFs is limited to only limited-scale wide-area network deployments because it cannot take the propagation latency of links into account, nor potential variations thereof. It also requires very high precision clock synchronization, which is uncommon in wide-area network equipment beyond mobile network fronthaul. See [[I-D.eckert-detnet-bounded-latency-problems](#)] for more details.

This specification introduces and utilizes an enhanced form of CQF where packets are tagged with a cycle identifier, and a limited number of cycles, e.g.: 3...7 are used to overcome these distance and clock synchronization limitations. Because this memo defines how to use the TC field of MPLS LSE as the tag to carry the cycle

identifier, it calls this scheme TC Tagged multiple buffer CQF (TC TCQF). See [[I-D.qiang-DetNet-large-scale-DetNet](#)] and [[I-D.dang-queueing-with-multiple-cyclic-buffers](#)] for more details of the theory of operations of TCQF. Note that TCQF is not necessarily limited to deterministic operations but could also be used in conjunction with congestion controlled traffic, but those considerations are outside the scope of this memo.

TCQF is likely especially beneficial when MPLS networks are designed to avoid per-hop, per-flow state even for traffic steering, which is the case for networks using SR-MPLS [[RFC8402](#)] for traffic steering of MPLS unicast traffic and/or BIER-TE [[I-D.ietf-bier-te-arch](#)] for tree engineering of MPLS multicast traffic. In these networks, it is specifically undesirable to require per-flow signaling to P-LSR solely for DetNet QoS because such per-flow state is unnecessary for traffic steering and would only be required for the bounded latency QoS mechanism and require likely even more complex hardware and manageability support than what was previously required for per-hop steering state (e.g. In RSVP-TE). Note that the DetNet architecture [[RFC8655](#)] does not include full support for this DiffServ model, which is why this memo describes how to use MPLS TC TCQF with the DetNet architecture per-hop, per-flow processing as well as without it.

## 2. Using TCQF with DetNet MPLS (informative)

This section gives an overview of how the operations of T-CQF relates to the DetNet architecture. We first revisit QoS with DetNet in the absence of T-CQF.

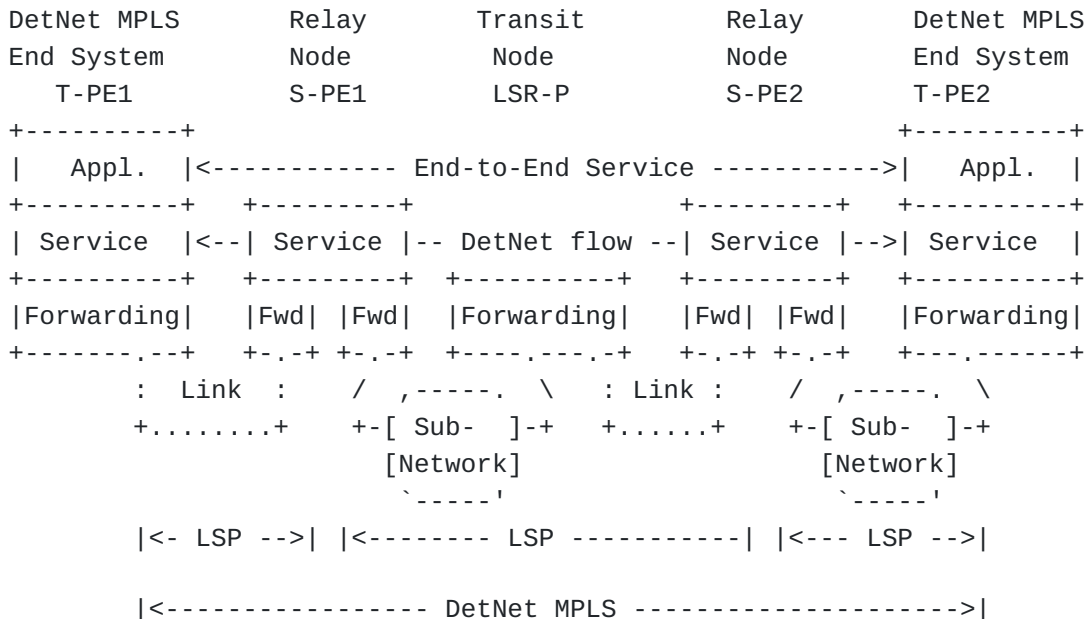


Figure 1: A DetNet MPLS Network

The above [Figure 1](#), is copied from [[RFC8964](#)], Figure 2, and only enhanced by numbering the nodes to be able to better refer to them in the following text.

Assume a DetNet flow is sent from T-PE1 to T-PE2 across S-PE1, LSR, S-PE2. In general, bounded latency QoS processing is then required on the outgoing interface of T-PE1 towards S-PE1, and any further outgoing interface along the path. When T-PE1 and S-PE2 know that their next-hop is a service LSR, their DetNet flow label stack may simply have the DetNet flows Service Label (S-Label) as its Top of Stack (ToS) LSE, explicitly indicating one DetNet flow.

On S-PE1, the next-hop LSR is not DetNet aware, which is why S-PE1 would need to send a label stack where the S-Label is followed by a Forwarding Label (F-Label), and LSR-P would need to perform bounded latency based QoS on that F-Label.

For bounded latency QoS mechanisms relying on per-flow regulator state, such as in [[TSN-ATS](#)], this requires the use of a per-detnet flow F-Label across the network from S-PE1 to S-PE2, for example through RSVP-TE [[RFC3209](#)] enhanced as necessary with QoS parameters matching the underlying bounded latency mechanism (such as [[TSN-ATS](#)]).

With TC TCQF, a sequence of LSR and DetNet service node implements TC TCQF, ideally from T-PE1 (ingress) to T-PE2 (egress). The ingress node needs to perform per-DetNet-flow per-packet "shaping" to assign each packet of a flow to a particular TCQF cycle. This ingress-edge-function is currently out of scope of this document (TBD), but would be based on the same type of edge function as used in CQF.

All LSR/Service node after the ingress node only have to map a received TCQF tagged DetNet packet to the configured cycle on the output interface, not requiring any per-DetNet-flow QoS state. These LSR/Service nodes do therefore also not require per-flow interactions with the controller plane for the purpose of bounded latency.

Per-flow state therefore is therefore only required on nodes that are DetNet service nodes, or when explicit, per-DetNet flow steering state is desired, instead of ingress steering through e.g.: SR-MPLS.

Operating TCQF per-flow stateless across a service node, such as S-PE1, S-PE2 in the picture is only an option. It is of course equally feasible to Have one TCQF domain from T-PE1 to S-PE2, start a new TCQF domain there, running for example up to S-PE2 and start another one to T-PE2.

A service node must act as an egress/ingress edge of a TCQF domain if it needs to perform operations that do change the timing of packets other than the type of latency that can be considered in configuration of TCQF (see [Section 7](#)).

For example, if T-PE1 is ingress for a TCQF domain, and T-PE2 is the egress, S-PE1 could perform the DetNet Packet Replication Function (PRF) without having to be a TCQF edge node as long as it does not introduce latencies not included in the TCQF setup and the controller plane reserves resources for the multitude of flows created by the replication taking the allocation of resources in the TCQF cycles into account.

Likewise, S-PE2 could perform the Packet Elimination Function without being a TCQF edge node as this most likely does not introduce any non-TCQF acceptable latency - and the controller plane accordingly reserves only for one flow the resources on the S-PE2->T-PE2 leg.

If on the other hand, S-PE2 was to perform the Packet Reordering Function (PRF), this could create large peaks of packets when out-of-order packets are released together. A PRF would either have to take care of shaping out those bursts for the traffic of a flow to again conform to the admitted CIR/PIR, or else the service node would have to be a TCQF egress/ingress, performing that shaping itself as an ingress function.

### **3. Data model and tag processing for MPLS TC TCQF (normative)**

```

module ietf-detnet-tcqf
  augment TBD
    +--rw tcqf-config
    |   +--rw cycles uint16
    |   +--rw cycle-time uint16
    |   +--rw cycle-clock-offset uint32
    |   +--rw tcqf-if-config* [oif-name]
    |       +--rw oif-name          if:interface-ref
    |       +--rw cycle-clock-offset int32
    |       +--rw tcqf-iif-cycle-map* [iif-name]
    |           +--rw iif-name      if:interface-ref
    |           +--rw iif-cycle-map* [iif-cycle]
    |               +--rw iif-cycle uint8
    |               +--rw oif-cycle uint8
    |
    +--rw tcqf-mpls-tc-tag* [name]
        +--rw name          if:interface-ref
        +--rw cycle* [cycle]
            +--rw cycle uint8
            +--rw tc      uint8

```

Figure 2: TCQF Data Model

tcqf-config is the router/LSR wide configuration of TCQF parameters, independent of the tagging of the method with which cycles are tagged on any interface. This YANG model represents a single TCQF domain, which is a set of interfaces acting both as ingress (iif) and egress (oif) interfaces, capable to forward TCQF packets amongst each other. When multiple independent instances or TCQF domains are used, they can have separate parameters.

cycles is the number of cycles used across all interfaces. router/LSR MUST support 3 and 4 cycles. To support interfaces with MPLS TC tagging, 7 or less cycles must be used.

The cycle time is cycle-time in units of micro-seconds. router/LSR MUST support configuration of cycle-times of 20,50,100,200,500,1000,2000 usec.

Cycles start at an offset of cycle-clock-offset in units of nsec as follows. Let clock1 be a timestamp of the local reference clock for TCQF, at which cycle 1 starts, then:

$$\text{cycle-clock-offset} = (\text{clock1} \bmod (\text{cycle-time} * \text{cycles}) )$$

The local reference clock is expected to be synchronized with the neighboring nodes. cycle-clock-offset can be configurable, or it may be derived from immutable properties of the implementation, in which case it is read-only.

tcqf-if-config is the optional per-interface configuration of TCQF parameters.

The cycle-clock-offset in tcqf-oif-config may be different from the router/LSR wide cycle-clock-offset, for example, when interfaces are on line cards with independently synchronized clocks, or when non-uniform ingress-to-egress propagation latency over a complex router/LSR fabric makes it beneficial to allow per-egress interface or line card configuration of cycle-clock-offset.

If cycle-clock-offset is unused and therefore the router/LSR wide cycle-clock-offset is used, the value MUST be -1. This is the only permitted negative number.

tcqf-iif-cycle-map is defining how to map the cycle iif-cycle of a packet received from an incoming interface (iif-name) once the LSR has determined that the packet needs to be sent to oif-name and sent with TCQF. The packet is then assigned to cycle oif-cycle on oif-name.

Note that all parameters so far allow for different methods of tagging the cycle in the packet across different interfaces and allowing TCQF to operate across them, even if future work would introduce different tagging methods than the following MPLS TC mapping.

tcqf-mpls-tc-tag defines the mapping of cycle number to MPLS TC tag. This mapping is configured for all interfaces that use MPLS TC tagging. When a packet is received with a ToS LSE indicating a TC for which there is a mapping to a cycle in tcqf-mpls-tc-tag, then this packet is assigned to the configured cycle.

If the packet is forwarded to another interface with tcqf configured, the cycle number derived from mapping the received ToS LSE TC field to the cycle number when receiving the packet will be mapped according to tcqf-oif-config after all label stack changes are applied and the packet is to be sent. If that outgoing interface is also using MPLS TC TCQF tagging, then the TC value of the ToS LSE will be rewritten according to the tcqf-mpls-tc-tag configuration of that outgoing interface.

tc in tcqf-mpls-tc-tag MUST NOT use values to be used for non-TCQF traffic, most commonly 0 for Best Effort (BE) traffic.

#### **4. TCQF with labels stack operations (normative)**

TCQF QoS as defined here is in the terminology of [[RFC3270](#)] a TC-Inferred-PSC LSP (E-LSP) behavior. Packets are determined to belong to the TCQF PSC solely based on the TC of the received packet.

Packets originated into the TCQF PSC on the ingress LSR are assumed for the purpose of this specification to be received from an internal interface for which the cycle mapping table on every interface is 1:1. This allows to distinguish the case of originated TCQF packets from those received from another LSR.

Note that this ingress mapping rule does not represent the shaping necessary on an ingress TCQF router. TBD.

Label swap in the case of LDP or RSVP-TE LSP, or label pop in the case of SR-MPLS traffic steering, or any other operation may result in a different label to become the ToS LSE. Whenever a packet has an associated TCQF cycle and is sent to an interface with TCQF, the cycle is mapped to that outgoing interfaces cycle space and the TC of the ToS LSE accordingly updated.

## **5. TCQF Pseudocode (normative)**

The following pseudocode restates the prior two section text in an algorithmic fashion. It uses the objects of the TCQF YANG data model defined in [Section 3](#).



```

tcqf = ietf-detnet-tcqf
void receive(pak) {

    // Receive side TCQF - remember cycle in
    // packet internal header
    iif = pak.context.iif
    if(tcqf.tcqf-if-config[iif]) { // TCQF enabled
        if(tcqf.tcqf-mpls-tc-tag[iif]) { // TC-TCQF
            pak.context.tcqf_cycle =
                map_tc2cycle(pak.mpls_header.lse[tos].tc,
                    tcqf.tcqf-mpls-tc-tag[iif])
        } else // other future encap/tagging options for TCQF
        }

    // Forwarding including any LSE operations
    oif = pak.context.oif = forward_process(pak)

    // ... optional DetNet PREOF functions here
    // ... if router is DetNet service node

    if(pak.context.tcqf_cycle && // non TCQF packets value is 0
        tcqf.tcqf-if-config[oif]) { // TCQF enabled
        // Map tcqf_cycle for iif to oif mapping table
        cycle = pak.context.tcqf_cycle =
            map_cycle(cycle,
                tcqf.tcqf-if-config[oif].tcqf-iif-cycle-map[[iif])

        if(tcqf.tcqf-mpls-tc-tag[iif]) { // TC-TCQF
            pak.mpls_header.lse[tos].tc =
                map_cycle2tc(cycle, tcqf.tcqf-mpls-tc-tag[oif])
        } else // other future encap/tagging options for TCQF

        tcqf_enqueue(pak, oif.cycleq[cycle])
    }
}

// Started when TCQF is enabled on an interface
void send_tcqf(oif) {
    cycle = 1
    cc = tcqf.tcqf-config.cycle-time *
        tcqf.tcqf-config.cycle-time
    o = tcqf.tcqf-config.cycle-clock-offset
    nextcyclestart = floor(tnow / cc) * cc + cc + o

    while(1) {
        while(tnow < nextcyclestart) { }
        while(pak = dequeue(oif.cycleq[cycle])) {
            send(pak)
        }
        cycle = (cycle + 1) mod tcqf.tcqf-config.cycles + 1
    }
}

```

```
        nextcyclestart += tcqf.tcqf-config.cycle-time
    }
}
```

Figure 3: TCQF Pseudocode

## 6. TCQF YANG Model (normative) TBD

TBD - according to [Section 3](#).

## 7. Computing cycle mappings (informative)

The cycle mapping is computed by the controller plane by taking at minimum the link, interface serialization and node internal forwarding latencies as well as the cycle-clock-offsets into account.

```
Router . 01
R1 . | cycle 1 | cycle 2 | cycle 3 | cycle 1 |
    .
    . ..... Delay D
    .
    . 01'
    . | cycle 1 |
Router . | cycle 1 | cycle 2 | cycle 3 | cycle 1 |
R2 . 02
```

CT = cycle-time

C = cycles

CC = CT \* C

01 = cycle-clock-offset router R1, interface towards R2

02 = cycle-clock-offset router R2, output interface of interest

01' = 01 + D

Figure 4: Calculation reference

Consider in {#Calc1} that Router R1 sends packets via C = 3 cycles with a cycle-clock offset of 01 towards Router R2. These packets arrive at R2 with a cycle-clock offset of 01' which includes through D all latencies incurred between releasing a packet on R1 from the cycle buffer until it can be put into a cycle buffer on R2: serialization delay on R1, link delay, non-CQF delays in R1 and R2, especially forwarding in R2, potentially across an internal fabric to the output interface with the sending cycle buffers.

$$A = (\text{ceil}((01' - 02) / CT) + C + 1) \bmod CC$$

$$\text{map}(i) = (i - 1 + A) \bmod C + 1$$

Figure 5: Calculating cycle mapping

{#Calc2} shows a formula to calculate the cycle mapping between R1 and R2, using the first available cycle on R2. In the example of

{#Calc1} with  $CT = 1$ ,  $(O1' - O2) \approx 1.8$ , A will be 0, resulting in  $map(1)$  to be 1,  $map(2)$  to be 2 and  $map(3)$  to be 3.

The offset "C" for the calculation of A is included so that a negative  $(O1 - O2)$  will still lead to a positive A.

In general, D will be variable [Dmin...Dmax], for example because of differences in serialization latency between min and max size packets, variable link latency because of temperature based length variations, link-layer variability (radio links) or in-router processing variability. In addition, D also needs to account for the drift between the synchronized clocks for R1 and R2. This is called the Maximum Time Interval Error (MTIE).

Let  $A(d)$  be A where  $O1'$  is calculated with  $D = d$ . To account for the variability of latency and clock synchronization,  $map(i)$  has to be calculated with  $A(Dmax)$ , and the controller plane needs to ensure that that  $A(Dmin)...A(Dmax)$  does cover at most  $(C - 1)$  cycles.

If it does cover C cycles, then C and/or CT are chosen too small, and the controller plane needs to use larger numbers for either.

This  $(C - 1)$  limitation is based on the understanding that there is only one buffer for each cycle, so a cycle cannot receive packets when it is sending packets. While this could be changed by using double buffers, this would create additional implementation complexity and not solve the limitation for all cases, because the number of cycles to cover [Dmin...Dmax] could also be  $(C + 1)$  or larger, in which case a tag of  $1...C$  would not suffice.

## 8. Security Considerations

TBD.

## 9. IANA Considerations

This document has no IANA considerations.

## 10. 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", 2017.

[I-D.dang-queuing-with-multiple-cyclic-buffers] Liu, B. and J. Dang, "A Queuing Mechanism with Multiple Cyclic Buffers", Work in Progress, Internet-Draft, draft-dang-queuing-with-multiple-cyclic-buffers-00, 22 February 2021, <<https://>

[www.ietf.org/archive/id/draft-dang-queuing-with-multiple-cyclic-buffers-00.txt](https://www.ietf.org/archive/id/draft-dang-queuing-with-multiple-cyclic-buffers-00.txt)>.

- [I-D.eckert-detnet-bounded-latency-problems] Eckert, T. and S. Bryant, "Problems with existing DetNet bounded latency queuing mechanisms", Work in Progress, Internet-Draft, draft-eckert-detnet-bounded-latency-problems-00, 12 July 2021, <<https://www.ietf.org/archive/id/draft-eckert-detnet-bounded-latency-problems-00.txt>>.
- [I-D.ietf-bier-te-arch] Eckert, T., Cauchie, G., and M. Menth, "Tree Engineering for Bit Index Explicit Replication (BIER-TE)", Work in Progress, Internet-Draft, draft-ietf-bier-te-arch-10, 9 July 2021, <<https://www.ietf.org/archive/id/draft-ietf-bier-te-arch-10.txt>>.
- [I-D.ietf-detnet-bounded-latency] Finn, N., Boudec, J. L., Mohammadpour, E., Zhang, J., Varga, B., and J. Farkas, "DetNet Bounded Latency", Work in Progress, Internet-Draft, draft-ietf-detnet-bounded-latency-07, 1 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-detnet-bounded-latency-07.txt>>.
- [I-D.qiang-DetNet-large-scale-DetNet] Qiang, L., Geng, X., Liu, B., Eckert, T., Geng, L., and G. Li, "Large-Scale Deterministic IP Network", Work in Progress, Internet-Draft, draft-qiang-DetNet-large-scale-DetNet-05, 2 September 2019, <<https://www.ietf.org/archive/id/draft-qiang-DetNet-large-scale-DetNet-05.txt>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <<https://www.rfc-editor.org/info/rfc3270>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <<https://www.rfc-editor.org/info/rfc8655>>.

**[RFC8964]**

Varga, B., Ed., Farkas, J., Berger, L., Malis, A., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: MPLS", RFC 8964, DOI 10.17487/RFC8964, January 2021, <<https://www.rfc-editor.org/info/rfc8964>>.

**[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/>>.

**Authors' Addresses**

Toerless Eckert  
Futurewei Technologies USA  
2220 Central Expressway  
Santa Clara, CA 95050  
United States of America

Email: [tte@cs.fau.de](mailto:tte@cs.fau.de)

Stewart Bryant  
University of Surrey ICS

Email: [s.bryant@surrey.ac.uk](mailto:s.bryant@surrey.ac.uk)