Workgroup: DETNET
Internet-Draft: draft-eckert-detnet-tcqf-01
Published: 6 November 2022
Intended Status: Standards Track
Expires: 10 May 2023
Authors: T. Eckert
Futurewei Technologies USA
S. Bryant
University of Surrey ICS
A. G. Malis
Malis Consulting
G. Li
Huawei Network Technology Laboratory

Deterministic Networking (DetNet) Data Plane - Tagged Cyclic Queuing and Forwarding (TCQF) for bounded latency with low jitter in large scale DetNets

Abstract

This memo specifies a forwarding method for bounded latency for Deterministic Networks. It uses cycle tagging of packets for cyclic queuing and forwarding with multiple buffers (TCQF). This memo standardizes tagging via the MPLS packet Traffic Class (TC) field for MPLS links and the IP/IPv6 DSCPfield for IP/IPv6 links. The short-hand for this mechanism is Tagged Cyclic Queuing and Forwarding (TCQF).

Target benefits of TCQF 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 TCQF 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

Table of Contents

1. Introduction (informative)

Cyclic Queuing and Forwarding (CQF), [IEEE802.1Qch], 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 cycle identifiers for a limited number of cycles (such as 3...7) and hop-by-hop forwarded through the use of per-cycle buffers. This multiple buffer forwarding overcome the distance and clock synchronization limitations of CQF. [I-D.qiang-DetNet-large-scale-DetNet] and [I-D.dang-queuing-with-multiple-cyclic-buffers] provide additional details about the background of TCQF. TCQF does not depend on other elements of [RFC8655], so it can also be used in otherwise non-deterministic IP or MPLS networks to achieve bounded latency and low jitter.

TCQF is likely especially beneficial when networks are architected 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, SRv6 [RFC8986] for traffic steeering of IPv6 unicast traffic and/or BIER-TE [I-D.ietf-bier-te-arch] for tree engineering of MPLS multicast traffic (using the TC and/or DSCP header fields of BIER packets according to [RFC8296]).

In these networks, it is specifically undesirable to require per-flow signaling to non-edge forwarders (such as P-LSR in MPLS networks) 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 (such as in RSVP-TE, [RFC4875]). Note that the DetNet architecture [RFC8655] does not include full support for this DiffServ model, which is why this memo describes how to use TCQF with the DetNet architecture per-hop, per-flow processing as well as without it.

2. Using TCQF in the DetNet Architecture and MPLS forwarding plane (informative)

This section gives an overview of how the operations of TCQF relates to the DetNet architecture. We first revisit QoS with DetNet in the absence of TCQF using an MPLS network as an example.

```
 DetNet MPLS       Relay       Transit         Relay       DetNet MPLS
 End System        Node         Node           Node        End System
    T-PE1          S-PE1        LSR-P          S-PE2       T-PE2
 +----------+                                             +----------+
 |   Appl.  |<------------ End-to-End Service ----------->|   Appl.  |
 +----------+   +---------+                 +---------+   +----------+
 | Service  |<--| Service |-- DetNet flow --| Service |-->| Service  |
 +----------+   +---------+  +----------+   +---------+   +----------+
 |Forwarding|   |Fwd| |Fwd|  |Forwarding|   |Fwd| |Fwd|   |Forwarding|
 +-------.--+   +-.-+ +-.-+  +----.---.-+   +-.-+ +-.-+   +---.------+
         :  Link  :    /  ,-----.  \   : Link :    /  ,-----.  \
         +........+    +-[ Sub-  ]-+   +......+    +-[ Sub-  ]-+
                           [Network]                   [Network]
                            `-----'                     `-----'
         |<- LSP -->| |<-------- LSP -----------| |<--- LSP -->|

         |<----------------- DetNet MPLS --------------------->|
```

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 (aka: per-flow packet scheduling), such as in [TSN-ATS], this requires the use of a per-detnet flow F-Labels across the network from S-PE1 to S-PE2. These could for for example be assigned/managed through RSVP-TE [RFC3209] enhanced as necessary with QoS parameters matching the underlying bounded latency mechanism (such as [TSN-ATS]).

With TCQF, a sequence of LSR and DetNet service node implements TCQF with MPLS TC, ideally from T-PE1 (ingress) to T-PE2 (egress). The ingress node needs to perform per-DetNet-flow per-packet "shaping"/"regulating" to assign each packet of a flow to a particular TCQF cycle. This is specified in Section 4.

All LSR/Service nodes 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 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 one 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 5.2).

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 TQCF 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. TCQF per-flow stateless forwarding (normative)

3.1. Configuration Data model and tag processing for MPLS TC tags

The following data model summarizes the configuration parameters as required for TCQF and discussed in further sections. 'tcqf' includes the parameters independent of the tagging on an interface. 'tcqf_*' describes the parameters for interfaces using MPLS TC and IP DSCP tagging.

```
# Encapsulation agnostic data
tcqf
+-- uint16 cycles
+-- uint16 cycle_time
+-- uint32 cycle_clock_offset
+-- if_config[oif] # Outgoing InterFace
    +-- uint32 cycle_clock_offset
    +-- cycle_map[iif] # Incoming InterFace
        +--uint8 oif_cycle[iif_cycle]

# MPLS TC tagging specific data
tcqf_tc[oif]
+--uint8 tc[oif_cycle]

# IP/IPv6 DSCP tagging specific data
tcqf_dscp[oif]
+--uint8 dscp[oif_cycle]
```

Figure 2: TCQF Configuration Data Model

3.2. Packet processing

This section explains the TCQF packet processing and through it, introduces the semantic of the objects in Figure 2

tcqf contains the router wide configuration of TCQF parameters, independent of the specific tagging mechanism on any interface. Any interface can have a different tagging method. This document uses the term router when it is irrelevant whether forwarding is for IP or MPLS packet, and the term Label Switched Router (LSR) to indicate MPLS is used, or IP router to indicate IP or IPv6 are used.

The model represents a single TQCF domain, which is a set of interfaces acting both as ingress (iif) and egress (oif) interfaces, capable to forward TCQF packets amongst each other. A router may have multiple TCQF domains each with a set of interfaces disjoint from those of any other TCQF domain.

tcqf.cycles is the number of cycles used across all interfaces in the TCQF domain. routers MUST support 3 and 4 cycles. To support interfaces with MPLS TC tagging, 7 or less cycles MUST be used across all interfaces in the CQF domain.

The unit of tcqf.cycle_time is micro-seconds. routers MUST support configuration of cycle-times of 20,50,100,200,500,1000,2000 usec.

Cycles start at an offset of tcqf.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:

tcqf.cycle_clock_offset = (clock1 mod (tcqf.cycle_time * tcqf.cycles))

The local reference clock of the LSR/router is expected to be synchronized with the neighboring LSR/router in TCQF domain. tcqf.cycle_clock_offset can be configurable by the operator, or it can be read-only. In either case will the operator be able to configure working TCQF forwarding through appropriately calculated cycle mapping.

tcqf.if_config[oif] is optional per-interface configuration of TCQF parameters. tcqf.if_config[oif].cycle_clock_offset may be different from tcqf.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. It may be configurable or read-only.

The value of -1 for tcqf.if_config[oif].cycle_clock_offset is used to indicate that the domain wide tcqf.cycle_clock_offset is to be used for oif. This is the only permitted negative number for this parameter.

When a packet is received from iif with a cycle value of iif_cycle and the packet is routed towards oif, then the cycle value (and buffer) to use on oif is tcqf.if_config[oif].cycle_map[iif].oif_cycle[iif_cycle]. This is called the cycle mapping and is must be configurable. This cycle mapping always happens when the packet is received with a cycle tag on an interface in a TCQF domain and forwarded to another interface in the same TCQF domain.

tcqf_tc[oif].tc[oif_cycle] defines how to map from the internal cycle number oif_cycle to an MPLS TC value on interface oif.

tcqf_tc[oif] MUST be configured, when oif uses MPLS. This oif_cycle <=> tc mapping is not only used to map from internal cycle number to MPLS TC tag when sending packets, but also to map from MPLS TC tag to the internal cycle number when receiving packets. Likewise, tcqf_dscp[oif] MUST be configured, when oif uses IP/IPv6.

This data model does not determine whether interfaces use MPLS or IP/IPv6 encapsulation. This is determined by the setup of the DetNet domain. A mixed use of MPLS and IP/IPv6 interfaces is possible with this data model, but at the time of writing this document not supported by DetNet.

3.3. TCQF with MPLS label stack operations

In the terminology of [RFC3270], TCQF QoS as defined here, is 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.

The internal cycle number SHOULD be assigned from the Top of Stack (ToS) MPLS label TC bits before any other label stack operations happens. On the egress side, the TC value of the ToS MPLS label SHOULD be assigned from the internal cycle number after any label stack processing.

With this order of processing, TCQF can support forwarding of packets with any label stack operations such as label swap in the case of LDP or RSVP-TE created LSP, Penultimate Hop Popping (PHP), or no label changes from SID hop-by-hop forwarding and/or SID/label pop as in the case of SR-MPLS traffic steering.

3.4. TCQF with IP operations

As how DetNet domains are currently assumed to be single administrative network operator domains, this document does not ask for standardization of the DSCP to use with TCQF. Instead, deployments wanting to use TCQF with IP/IPv6 encapsulation need to assign within their domain DSCP from the xxxx11 "EXP/LU" Codepoint space according to [RFC2474], Section 6. This allows up to 16 DSCP for intradomain use.

3.5. TCQF Pseudocode (normative)

The following pseudocode restates the forwarding behavior of Section 3 in an algorithmic fashion as pseudocode. It uses the objects of the TCQF configuration data model defined in Section 3.1.

```
void receive(pak) {
  // Receive side TCQF - retrieve cycle of received packet
  // from packet internal header
  iif = pak.context.iif
  if (tcqf.if_config[iif]) { // TCQF enabled on iif
    if (tcqf_tc[iif]) {      // MPLS TCQF enabled on iif
      tc = pak.mpls_header.lse[tos].tc
      pak.context.tcqf_cycle = map_tc2cycle( tc, tcqf_tc[iif])
    } else
    if (tcqf_dscp[iif]) {      // IP TCQF enabled on iif
      dscp = pak.ip_header.dscp
      pak.context.tcqf_cycle = map_dscp2cycle( dscp, tcqf_dscp[iif])
    } else // ... other encaps
  }
  forward(pak);
}

// ... Forwarding including any label stack operations

void forward(pak) {
  oif = pak.context.oif = forward_process(pak)

  if(ingres_flow_enqueue(pak))
    return // ingress packets are only enqueued here.

  if(pak.context.tcqf_cycle) // non TCQF packets cycle is 0
    if(tcqf.if_config[oif]) {    // TCQF enabled on OIF
      // Map tcqf_cycle iif to oif - encap agnostic
      cycle = pak.context.tcqf_cycle
            = map_cycle(cycle,
              tcqf.if_config[oif].cycle_map[[iif])

      // MPLS TC-TCQF
      if(tcqf.tc[oif]) {
        pak.mpls_header.lse[tos].tc = map_cycle2tc(cycle, tcqf_tc[oif])
      } else
      // IP TCQF enabled on iif
      if (tcqf_dscp[oif]) {
        pak.ip_header.dscp = map_cycle2dscp(cycle, tcqf_dscp[oif])
      } // else...  other future encap/tagging options for TCQF

      tcqf_enqueue(pak, oif.cycleq[cycle])
      return
    } else {
      // Forwarding of egress TCQF packets [1]
    }
  }
  // ... non TCQF OIF forwarding [2]
}
```

```
// Started when TCQF is enabled on an interface
// dequeues packets from oif.cycleq
// independent of encapsulation
void send_tcqf(oif) {
  cycle = 1
  cc =  tcqf.cycle_time *
        tcqf.cycle_time
  o =   tcqf.cycle_clock_offset
  nextcyclestart = floor(tnow / cc) * cc + cc + o

  while(1) {
    ingres_flow_2_tcqf(oif,cycle)
    while(tnow < nextcyclestart) { }
    while(pak = dequeue(oif.cycleq(cycle)) {
      send(pak)
    }
    cycle = (cycle + 1) mod tcqf.cycles + 1
    nextcyclestart += tcqf.cycle_time
  }
}
```

Figure 3: TCQF Pseudocode

Processing of ingress TCQF packets is performed via ingres_flow_enqueue(pak) and ingres_flow_2_tcqf(oif,cycle) as explained in Section 4.2.

Processing of egres TCQF packet is out-of-scope. It can performed by any non-TCQF packet forwarding mechanism such as some strict priority queuing in [2], and packets could accordingly be marked with an according packet header traffic class indicator for such a traffic class in [1].

4. TCQF Per-flow Ingress forwarding (normative)

Ingress flows in the context of this text are packets of flows that enter the router from a non-TCQF interface and need to be forwarded to an interface with TCQF.

In the most simple case, these packets are sent by the source and the router is the first-hop router. In another case, the routers ingress interface connects to a hop where the previous router(s) did perform a different bounded latency forwarding mechanism than TCQF.

4.1. Ingress Flows Configuration Data Model

```
# Extends above defined tcqf
tcqf
...
| Ingress Flows, see below (TBD:
+-- iflow[flowid]
    +-- uint32 csize # in bits
```

Figure 4: TCQF Ingress Configuration Data Model

The data model shown in Figure 4 expands the tcqf data model from Figure 2. For every DetNet flow for which this router is the TCQF ingress, the controller plane has to specify a maximum number of bits called csize (cycle size) that are permitted to go into each individual cycle.

Note, that iflow[flowid].csize is not specific to the sending interface because it is a property of the DetNet flow.

4.2. Ingress Flows Pseudocode

When a TCQF ingress is received, it first has to be enqueued into a per-flow queue. This is necessary because the permitted burst size for the flow may be larger than what can fit into a single cycle, or even into the number of cycles used in the network.

```
bool ingres_flow_enqueue(pak) {
  if(!pak.context.tcqf_cycle &&
      flowid = match_detnetflow(pak)) {
    police(pak) // according to RFC9016 5.5
    enqueue(pak, flowq[oif][flowid])
    return true
  }
  return false
}
```

Figure 5: TCQF Ingress Enqueue Pseudocode

ingres_flow_enqueue(pak) as shown in Figure 5 performs this enqueuing of the packet. Its position in the DetNet/TCQF forwarding code is shown in Figure 3.

police(pak): If the router is not only the TCQF ingress router, but also the first-hop router from the source, ingres_flow_enqueue(pak) will also be the place where policing of the flows packet according to the Traffic Specification of the flow would happen - to ensure that packets violating the Traffic Specification will not be forwarded, or be forwarded with lower priority (e.g.: as best effort). This policing and resulting forwarding action is not specific to TCQF and therefore out of scope for this text. See [RFC9016], section 5.5.

```
void ingres_flow_2_tcqf(oif, cycle) {
  foreach flowid in flowq[oif][*] {
    free = tcqf.iflow[flowid].csize
    q = flowq[oif][flowid]
    while(notempty(q) &&
          (l = head(q).size) <= free) {
      pak = dequeue(q)
      free -= l
      tcqf_enqueue(pak, oif.cycleq[cycle])
    }
  }
}
```

Figure 6: TCQF Ingress Pseudocode

ingres_flow_2_tcqf(oif, cycle) as shown in Figure 6 transfers ingress DetNet flow packets from their per-flow queue into the queue of the cycle that will be sent next. The position of ingres_flow_2_tcqf() in the DetNet/TCQF forwarding code is shown in Figure 3.

5. Implementation, Deployment, Operations and Validation considerations (informative)

5.1. High-Speed Implementation

High-speed implementations with programmable forwarding planes of TCQF packet forwarding require Time-Gated Queues for the cycle queues, such as introduced by [IEEE802.1Qbv] and also employed in CQF [IEEE802.1Qch].

Compared to CQF, the accuracy of clock synchronization across the nodes is reduced as explained in Section 5.2 below.

High-speed forwarding for ingress packets as specified in Section 4 above would require to pass packets first into a per-flow queue and then re-queue them into a cycle queue. This is not ideal for high speed implementations. The pseudocode for ingres_flow_enqueue() and ingres_flow_2_tcqf(), like the rest of the pseudocode in this document is only meant to serve as the most compact and hopefully most easy to read specification of the desired externally observable behavior of TCQF - but not as a guidance for implementation, especially not for high-speed forwarding planes.

High-speed forward could be implemented with single-enqueueing into cycle queues as follows:

Let B[f] be the maximum amount of data that the router would need to buffer for ingress flow f at any point in time. This can be calculated from the flows Traffic Specification. For example, when using the parameters of [RFC9016], section 5.5.

```
B[f] <= MaxPacketsPerInterval*MaxPayloadSize*8

maxcycles = max( ceil( B[f] / tcqf.iflow[f].csize) | f)
```

Maxcycles is the maximum number of cycles required so that packets from all ingress flows can be directly enqueued into maxcycles queues. The router would then not cycle across tcqf.cycles number of queues, but across maxcycles number of queues, but still cycling across tcqf.cycles number of cycle tags.

Calculation of B[f] and in result maxcycles may further be refined (lowered) by additionally known constraints such as the bitrates of the ingress interface(s) and TCQF output interface(s).

5.2. Controller plane computation of cycle mappings

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   . O1
 R1      . | cycle 1 | cycle 2 | cycle 3 | cycle 1 |
         .    .
         .     ............... Delay D
         .                    .
         .                    O1'
         .                     | cycle 1 |
Router   .   | cycle 1 | cycle 2 | cycle 3 | cycle 1 |
  R2     .   O2

CT  = cycle_time
C   = cycles
CC  = CT * C
O1  = cycle_clock_offset router R1, interface towards R2
O2  = cycle_clock_offset router R2, output interface of interest
O1' = O1 + D
```

Figure 7: Calculation reference

Consider in Figure 7 that Router R1 sends packets via C = 3 cycles with a cycle_clock offset of O1 towards Router R2. These packets arrive at R2 with a cycle_clock offset of O1' 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 = ( ceil( ( O1' - O2 ) / CT) + C + 1) mod CC
map(i) = (i - 1 + A) mod C + 1
```

Figure 8: Calculating cycle mapping

Figure 8 shows a formula to calculate the cycle mapping between R1 and R2, using the first available cycle on R2. In the example of Figure 7 with CT = 1, (O1' - O2) =~ 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.

5.3. Link speed and bandwidth sharing

TCQF hops along a path do not need to have the same bitrate, they just need to use the same cycle time. The controller plane has to then be able to take the TCQF capacity of each hop into account when admitting flows based on their Traffic Specification and TCQF csize.

TCQF does not require to be allocated 100% of the link bitrate. When TCQF has to share a link with other traffic classes, queuing just has to be set up to ensure that all data of a TCQF cycle buffer can be sent within the TCQF cycle time. For example by making the TCQF cycle queues the highest priority queues and then limiting their capacity through admission control to leave time for other queues to be served as well.

5.4. Validation

[LDN] describes an experimental validation of TCQF with high-speed forwarding hardware and provides further details on the mathematical models.

6. Security Considerations

TBD.

7. IANA Considerations

This document has no IANA considerations.

8. Changelog

[RFC-editor: please remove]

Initial draft name: draft-eckert-detnet-mpls-tc-tcqf

00

Initial version

01

Added new co-author.

Changed Data Model to "Configuration Data Model",

and changed syntax from YANG tree to a non-YANG tree, removed empty section targeted for YANG model. Reason: the configuration parameters that we need to specify the forwarding behavior is only a subset of what likely would be a good YANG model, and any work to define such a YANG model not necessary to specify the algorithm would be scope creep for this specification. Better done in a separate YANG document. Example additional YANG aspects for such a document are how to map parameters to configuration/operational space, what additional operational/monitoring parameter to support and how to map the YANG objects required into various pre-existing YANG trees.

Improved text in forwarding section, simplified sentences, used simplified configuration data model.

02

Refresh

03

Added ingress processing, and further implementation considerations.

New draft name: draft-eckert-detnet-tcqf

00

Added text for DSCP based tagging of IP/IPv6 packets, therefore changing the original, MPLS-only centric scope of the document, necessitating a change in name and title.

This was triggered by the observation of David Black at the IETF114 DetNet meeting that with DetNet domains being single administrative domains, it is not necessary to have standardized (cross administrative domain) DSCP for the tagging of IP/IP6 packets for TCQF. Instead it is sufficient to use EXP/LU DSCP code space and assignment of these is a local matter of a domain as is that of TC values when MPLS is used. Standardized DSCP in the other hand would have required likely work/oversight by TSVWG.

In any case, the authors feel that with this insight, there is no need to constrain single-domain definition of TCQF to only MPLS, but instead both MPLS and IP/IPv6 tagging can be easily specified in this one draft.

01

Added new co-author.

9. References

9.1. Normative References

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[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>.

[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>.

9.2. Informative References

[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>.

[I-D.eckert-detnet-bounded-latency-problems] Eckert, T. 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. T., Menth, M., and G. Cauchie, "Tree Engineering for Bit Index Explicit Replication (BIER-TE)", Work in Progress, Internet-Draft, draft-ietf-bier-te-arch-13, 25 April 2022, <https://www.ietf.org/archive/id/draft-ietf-bier-te-arch-13.txt>.

[I-D.ietf-detnet-bounded-latency] Finn, N., Le Boudec, J., Mohammadpour, E., Zhang, J., and B. Varga, "DetNet Bounded Latency", Work in Progress, Internet-Draft, draft-ietf-detnet-bounded-latency-10, 8 April 2022, <https://www.ietf.org/archive/id/draft-ietf-detnet-bounded-latency-10.txt>.

[I-D.qiang-DetNet-large-scale-DetNet] Qiang, L., Geng, X., Liu, B., Eckert, T. 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>.

[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", 2015.

[IEEE802.1Qch] 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.

[LDN] Liu, B., Ren, S., Wang, C., Angilella, V., Medagliani, P., Martin, S., and J. Leguay, "Towards Large-Scale Deterministic IP Networks", IEEE 2021 IFIP Networking Conference (IFIP Networking), doi 10.23919/IFIPNetworking52078.2021.9472798, 2021.

[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>.

[RFC4875] Aggarwal, R., Ed., Papadimitriou, D., Ed., and S. Yasukawa, Ed., "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, DOI 10.17487/RFC4875, May 2007, <https://www.rfc-editor.org/info/rfc4875>.

[RFC8296] Wijnands, IJ., Ed., Rosen, E., Ed., Dolganow, A., Tantsura, J., Aldrin, S., and I. Meilik, "Encapsulation for Bit Index Explicit Replication (BIER) in MPLS and Non-MPLS Networks", RFC 8296, DOI 10.17487/RFC8296, January 2018, <https://www.rfc-editor.org/info/rfc8296>.

[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>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

[RFC9016] Varga, B., Farkas, J., Cummings, R., Jiang, Y., and D. Fedyk, "Flow and Service Information Model for Deterministic Networking (DetNet)", RFC 9016, DOI 10.17487/RFC9016, March 2021, <https://www.rfc-editor.org/info/rfc9016>.

[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

Stewart Bryant
University of Surrey ICS

Email: s.bryant@surrey.ac.uk

Andrew G. Malis
Malis Consulting

Email: agmalis@gmail.com

Guangpeng Li
Huawei Network Technology Laboratory

Email: liguangpeng@huawei.com