DETNET
Internet-Draft
Intended status: Standards Track
Expires: 10 May 2023

T. Eckert
Futurewei Technologies USA
S. Bryant
University of Surrey ICS
A. G. Malis
Malis Consulting
G. Li
Huawei Network Technology Laboratory

6 November 2022

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

draft-eckert-detnet-tcqf-01

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

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 10 May 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

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.

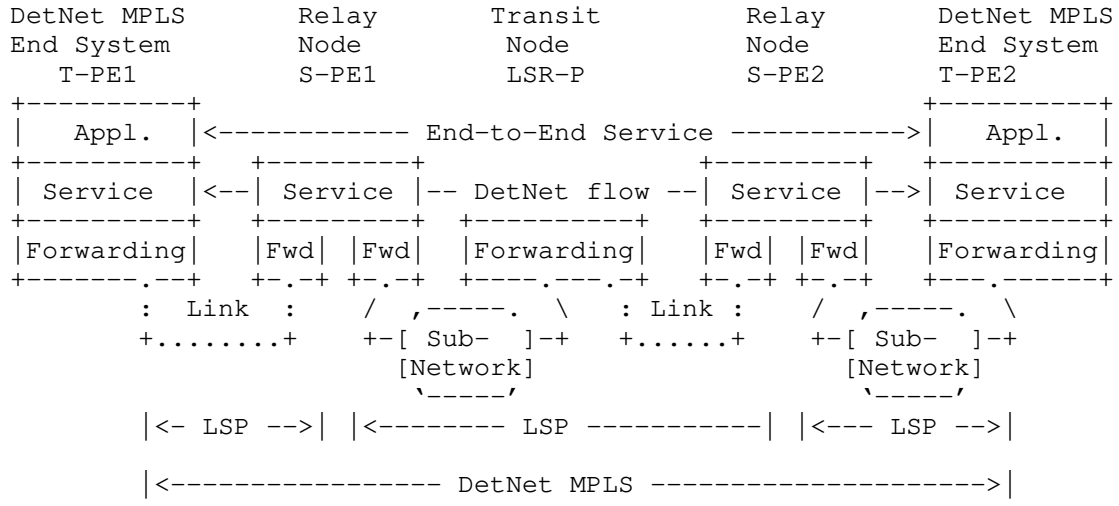


Figure 1: A DetNet MPLS Network

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

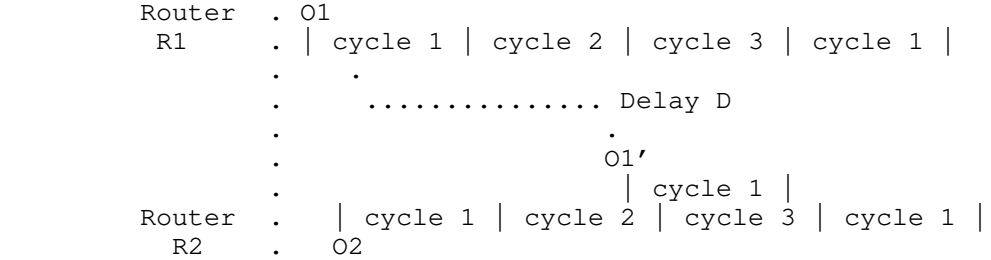
maxcycles = max(ceil(B[f] / tcqf.iflow[f].csize) | f)

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.



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

Deterministic Networking Working Group
Internet-Draft
Intended status: Informational
Expires: 6 June 2023

P. Liu
China Mobile
Y. Li
Huawei
T. Eckert
Futurewei Technologies USA
Q. Xiong
ZTE Corporation
J. Ryoo
ETRI
S. Zhu
New H3C Technologies
X. Geng
Huawei

3 December 2022

Requirements for Large-Scale Deterministic Networks
draft-ietf-detnet-scaling-requirements-00

Abstract

Aiming at the large-scale deterministic network with long hops, large per-hop time variation, great number of flows and/or multiple domains without the same time source, this document describes the technical and operational requirements when the different deterministic levels of applications co-exist and are transported. This document also describes the corresponding Deterministic Networking (DetNet) data plane enhancement requirements.

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 6 June 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Packet networks are evolving from bandwidth-guaranteed Quality of Service (QoS) to latency-guaranteed QoS that guarantees bounded latency and definite latency. Bounded latency and definite latency can be further understood as in-time delivery, in which a packet arrives without exceeding a predetermined time, and on-time delivery, in which a packet arrives at a predetermined time, respectively. In addition, network survivability, which typically guarantees traffic recovery within 50 ms in the event of a network failure, is evolving to a level that guarantees lossless recovery. In order to realize the evolution of QoS and network survivability of these networks, Time-Sensitive Networking (TSN) technology and Deterministic Networking (DetNet) technology are considered to be essential.

TSN is a set of standards developed by the IEEE 802.1 TSN Task Group (TG) [IEEE802.1TSN] and specifies mechanisms and protocols necessary to realize highly available IEEE 802.1 networks with bounded latency to carry time-sensitive, real-time application traffic.

DetNet, of which architecture is defined in RFC 8655 [RFC8655], provides a capability to carry specified unicast or multicast data flows for real-time applications with extremely low data loss rates and bounded latency under a single administrative control or within a closed group of administrative control. The overall framework for DetNet data plane is provided in [RFC8938], and various documents on different data plane technologies and their interworking technologies to extend the service range of data that TSN intends to deliver to the IP (Internet Protocol) and MPLS (Multi-Protocol Label Switching) networks have been standardized.

Since TSN and DetNet were proposed, application use cases have always been one of the hottest topics. After years of development, TSN has been used in several industries, and has enough public awareness of the industry for its scope. DetNet also has done a lot of work and the standards are mature, and people become concerned about how to meet deterministic service demand in large-scale networks.

In this document we define a large-scale DetNet network as a network that requires DetNet solutions for typically one or more of the following key attributes:

* There is relaxed clock synchronization or no clock synchronization in different domains.

* The end to end path is a combination of short and long distance hops.

* There are various transmission rate supported at the different ports and on different network node.

* There are a large number of flows which may has different level demands of DetNet service accrossing multi-domains.

* The topology change and failures of link might be more common.

* The mechanisms used to ensure bounded latency (e.g. queuing mechanism) may be multiple or have different configuration/parameter in multi-domains.

Such domains are normally within a single administrative control network or multiple cooperating administrative networks within a closed group of administrative control [RFC8655]. However they may belong to different AS domains and be controlled by multiple peering or cascaded controllers, and at the same time they may not have the same time clock source. Maintaining per flow status becomes impractical in the large scale network. Aggregation and disaggregation of flows take place at the boundaries of Detnet domains as well as within a Detnet domain with various rules. The flow identification and packet treatment should take care of one or combined changes introduced by the large-scale network.

Based on the defination and characteristics above, this document describes requirements for large-scale deterministic networks which demands the enhancement based on the existing bounded latency mechanisms and the corresponding data plane to ensure the detnet service for single administrative network or multiple (cooperating) administrative networks that defined in the detnet charter. The deterministic network for open internet is not within current scope.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

While [RFC2119] and [RFC8174] describe interpretations of these key words in terms of protocol specifications and implementations, they are used in this document to describe technical and operational requirements to realize large-scale deterministic networks.

3. Technical Requirements in Large-Scale Deterministic Networks

Due to the attributes described in Introduction Section, the corresponding technical requirements should be considered.

3.1. Tolerate Time Asynchrony

The large-scale network may span over multiple networks with one or more cooperating administrative domains. There are many types of network nodes in the multiple domains which may introduce disparate local time variation, which require the tolerance of time asynchrony.

3.1.1. Support Asynchronous Clocks Across Domains

A large-scale network may span over multiple networks with one or more administrative domains. One of DetNet's objectives is to stitch TSN islands together. All devices inside a TSN domain are time-synchronized, and most of TSN technologies rely on precise time synchronization [IEEE802.1Qbv][IEEE802.1Qch][IEEE802.1Qav]. However, different TSN islands may have different clocks which are not synchronized as shown in Figure 2, where the time difference of two TSN domains is D. DetNet needs to connect these two TSN domains together and provide end-to-end deterministic latency service. The mechanism adopted by a large-scale deterministic network MUST be prepared to cope with non-synced TSN domains. This can be done, for example, by putting extra buffer space at the ingress of a new domain, increasing the dead time as a guard band, or using some timing compensation mechanism. This document does not intend to list all the potential ways.

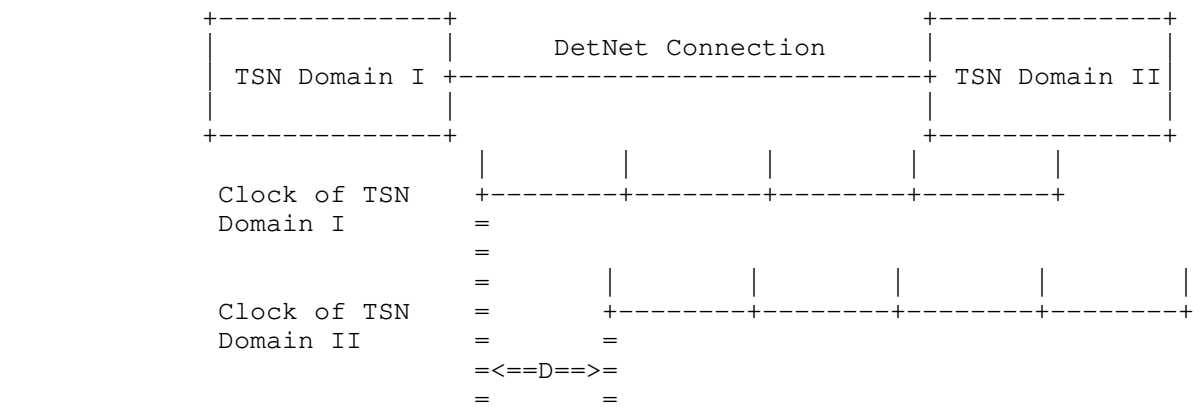


Figure 1: Clock asynchrony between two TSN islands

3.1.2. Tolerate Clock Jitter & Wander within a Clock Synchronous Domain

Within a single time synchronization domain, different clock accuracy is expected, for example the crystal oscillator in Ethernet is specified at 100 ppm [Fast-Ethernet-MII-clock], Synchronous Ethernet (SyncE) can achieve 50 ppb [G.8262], and more precise time synchronization [G.8273] is expected in 5G mobile backhaul. The clocks experience different jitter and wander. It may cause different level of asymmetry of the path. The large-scale networks SHOULD be able to recover or absorb such time variance within a domain and across multiple domains.

3.1.3. Provide Mechanisms not Requiring Full Time Synchronization

Some networks like mobile backhaul use frequency synchronization, such as SyncE, instead of the strict time synchronization. It is usually hard to achieve the full time synchronization in large-scale networks when considering the size of the network topology. It is desired that the same deterministic performance in term of the bounded latency and jitter SHOULD be achieved when full time synchronization is not available, that is to say, when only partial synchronization (SyncE is one of the examples) is in use.

3.1.4. Support Asynchronization based Methods

There are a large number of traffic flows in a large-scale network and some of them are acyclic. Asynchronization based methods can meet the requirements of those traffic flows. Moreover, The mechanisms not requiring the time and/or frequency synchronization eliminate the hardware cost and difficulty at the network nodes. [IEEE802.1Qcr] conceptually uses per-flow based asynchronous shaper

to achieve bounded latency. The effectiveness of the per-flow based asynchronous shaper can be proven through mathematical analysis. It can naturally tolerate the time variance, but it exhibits the concerns of per-flow state buffer management as shown in [I-D.eckert-detnet-bounded-latency-problems]. When it is in use, the requirement in Section 4.3 SHOULD be carefully met.

3.2. Support Large Single-hop Propagation Latency

In a large-scale network, a single hop distance is enough to generate large latency. The speed of optical transmission in fiber is 200 km/ms. Thus, the propagation delay of a single hop can be in the order of a few milliseconds. It is much greater than that of a LAN, and introduces impacts on queuing mechanisms, such as cyclic or time aware scheduling method. So the queuing mechanism for LAN networks needs to be extended, such as considering the propagation latency when setting the period in both time synchronization or frequency synchronization based methods, or setting the extra buffer in the asynchronization based methods, to meet the requirements of deterministic forwarding between the network nodes.

Here, we use an example to describe the influence of Large Single-hop Propagation Latency on cycle based methods, but not to specify any solution. For a cyclic based method, suppose a large-scale network wants to keep using the simple cycle mapping relationship, however the link distance between two nodes is longer. Moreover, a downstream node may have many upstream nodes each with different link propagation delays (e.g., 9 us, 10 us, 11 us, 15 us and 20 us). In order to absorb the longest link propagation delay, the length of cycle must be set to at least 20 us. However, since packet's arrival time varies within the receiving cycle, larger cycle length means larger delay variance.

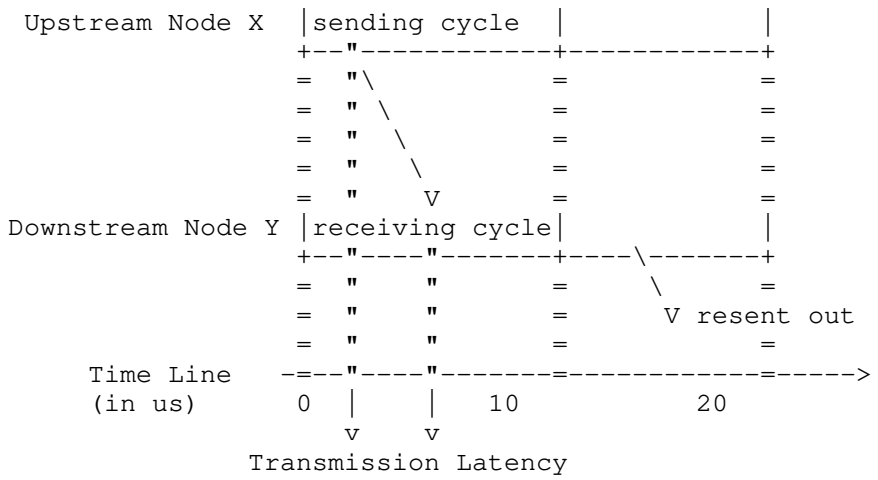


Figure 2: The influence of transmission latency on a cyclic method

3.3. Accommodate the Higher Link Speed

A large-scale network normally uses higher speed links, especially for its backbone. Current deterministic mechanisms used in a local network is usually deployed in link speed of 10 Mbps or 1 Gbps, or possibly 10 Gbps. The data rates of 10G, 100G, 400G and even higher are commonly used in wide area networks. With the increasing of the data rate, the network scheduling cycle can be reduced if the same amount of the data is required to be sent each cycle for each application. Or more data can be sent if the network cycle time remains the same. For the former, it requires the more precise time control (e.g. cycle in the order of a few microseconds or sub-microseconds) for the input stream gate and the timed output buffer. For the latter, more buffer space is required which imposes more complex buffer or queue management and larger memory consumption.

Another aspect to consider is the aggregation of the flows. In the large-scale network, the number of flows can be hundreds or tens of thousands. They can be aggregated into a small number of deterministic path or tunnels. It is practical to have a few flow-based or aggregated-flow based status in the local network. But in higher speed and larger scale networks, it is hardly feasible. If [IEEE802.1Qcr] is in use, it requires more buffers comparing to the other full/partial time synchronized mechanisms. Therefore, it requires optimizations to support higher link speeds.

3.4. Be Scalable to The Large Number of Flows

The large-scale network may have more traffic flows, which has different levels demand of detnet service, and access in/ leave out the detnet more irregular. The deterministic latency forwarding mechanisms MUST scale to networks of significant size with the massive traffic flows.

There are a large number of flows which may has different demands of DetNet service in large-scale deterministic network. [RFC8578] provides various use cases and their requirements in the areas of industry, electricity, buildings, etc. Some of them clearly specify the requirements for latency and jitter, while some others do not for the jitter. Different types of users have different demands, just as a network provider provides different network services for personal business or enterprise business.

One kind has critical SLA requirement, such as remote control or cloud Programmable Logic Controller (PLC) of manufacturing and differential protection of electricity. If these services exceed the boundaries of latency and jitter, it will bring property losses and security risks, so they cannot tolerate with any non-deterministic situation and can pay more on the network service. Another kind has relatively loose levels of SLA requirement, such as cloud gaming, cloud VR and online meeting for "consumer" networks. The users of these applications hope to have a better network experience, but they can tolerate it to a certain extent. For instance, exceeding the upper boundary of latency within a small probability is acceptable. Those different applications expect different kind of solutions, which are related to the cost more or less. For strict deterministic services, strict technologies need to be used, and all network devices may need to be upgraded. For non-strict deterministic services, it may only be necessary to upgrade some network devices (maybe edge nodes) or share corresponding network resources.

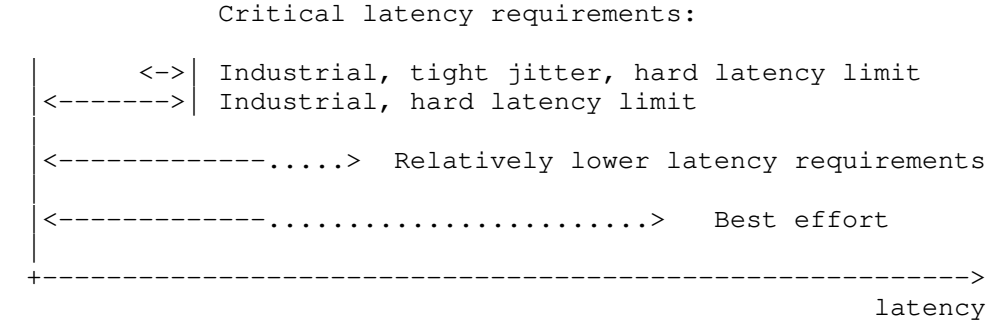


Figure 3: Different levels of application requirements

Besides, It is almost impossible to identify individual IP flows at the DetNet data plane because of the large overhead and resource reservation for a massive number of flows. DetNet allows the leverage of the flow aggregation. With the large scaling of the network, proper provision at the control plane to accommodate such higher aggregation is required. Individual flows may join and exit the aggregated flow rapidly which causes the dynamic in identification of the aggregated DetNet flow. The wildcards and value ranges used in the identification may have to change in order to ensure the aggregated flows have compatible deterministic characteristics.

The micro-burst will happen more often due to the massive traffic flows, so some methods to decrease it are needed. [I-D.du-detnet-layer3-low-latency] introduces a reference method requiring a scalable buffer to adjust the speed of sending the packets, so as to keep a uniform transmission rate, and it also support the flow aggregation. Moreover, the edge shaping based solution to reduce the micro-burst may also work to some extent.

3.5. Tolerate Failures of Links or Nodes and Topology Changes

The large-scale network may have more network devices, and the increase or decrease of network devices in large-scale networks is more frequent than that in LANs. A simple use case to understand is ultra-low-latency (public) 5G transport networks, which would require DetNet extend to every 5G base station. For some network operators, their networks may need to connect to ~100 K base stations (serving multiple mobile networks operators), and this number will only increase with 5G.

One the one hand, the numerous devices may lead to more network link failures. Path switching or re-convergence of routing will cause high latency of packet loss and retransmission, which is usually in seconds before the network becomes stable again. It is necessary to support certain mechanisms to adapt to failures of links or nodes and topology changes.

One the other hand, the change of the number of devices may affect the implementation and adjustment of deterministic network mechanism, such as the topology discovery, queuing mechanism and packet replication and elimination. For instance, The full disjoint paths when implementing the Packet Replication, Elimination, and Ordering Functions (PREOF) gives a better chance of survival when one of the nodes or links in the path fails. At the same time, it brings the challenges of finding paths with similar distance and/or number of hops so that there is enough buffer space to absorb the latency difference caused by different paths when the scale is large. So, a method is expected to support flexible planning of multiple paths and provide a solid foundation for the implementation of PREOF.

3.6. Support Enhancement of Queuing Mechanisms

3.6.1. Support Configuration of Multiple Queuing Mechanisms

It is required to provide diversified deterministic service for various applications in a large-scale network and to support the corresponding diversified queuing mechanisms (possibly at multiple DetNet QoS levels). Different queuing mechanisms can provide different levels of latency, jitter and other guarantees, and there may be situations where a network device provides multiple queuing mechanisms at the same time. For example, a network aggregation device may use the mechanisms specified in [IEEE802.1Qbv] and [IEEE802.1Qcr], and other mechanisms to forward traffic to different paths at the same time. By providing a variety of queuing mechanisms to meet diversified deterministic service requirements, compared with small scale environment, this demand is particularly prominent in large-scale networks. For instance, there may be more than eight queues or sub-queues to support more complicated queuing mechanisms comparing with the eight traffic classes in TSN enabled networks.

Accordingly, the configuration for multiple queuing mechanisms is complicated in large-scale deterministic networks and MUST support the unified or simplified scheduling and management of multiple queuing mechanisms. For example, in the distributed scenario with no controller, the related information of the queuing mechanisms could be advertised among the domain , including the types and related algorithms, queue forwarding capability with different levels of latency and jitter guarantees , etc. In the centralized scenario, the queuing mechanisms and other information could be reported to the controller to build a deterministic network resource topology pool for path calculation. In addition, for refined management of forward resources and providing resource assurance for deterministic forwarding when establishing/ deleting sessions, it is required to establish unified mechanisms on quantification and measurement of resources associated with interfaces and queues.

3.6.2. Support Queuing Mechanisms Switchover Crossing Multi-domains

In large-scale deterministic networks, it may across multiple network domains and adopt a variety of different queuing mechanisms within each domain. It is required to support the inter-domain deterministic mechanism at the inter-domain boundary nodes such as the priority redefinition and rescheduling of queues to achieve the end-to-end latency, bounded jitter and packet loss ratio.

Moreover, changing from one queuing mechanism to another may generate additional end-to-end latency and/or jitter which should be taken into consideration,because the different scheduling granularities or phase differences between the two domains requires flexible flow aggregation and queue stitching function. For example, when a flow is forwarded across multiple network domains based on different queuing mechanisms, such as a time synchronous Qbv mechanism [IEEE802.1Qbv] and an asynchronous Qcr mechanism [IEEE802.1Qcr], a collaboration mechanism crossing multi-domains MUST be considered, such as increasing the buffer of inter-domain devices to provide enough adjustment space for the flow to cross different queuing mechanisms, the expected method of jitter compression to reduce the coupling between two domains' queuing mechanisms, or the additional traffic shaping solutions to make the traffic smooth, so as to provide end-to-end deterministic services across multiple network domains.

4. Data Plane Enhancement Requirements

According to [RFC8938], the DetNet data plane can provide or carry two metadata in MPLS and IP data planes: Flow-ID and sequence number. The Flow-ID could be used for identification of the DetNet flow or aggregate flow, and the sequence number could be used for PREOF for each DetNet flow. The Flow-ID is used by both the service and forwarding sub-layers, but the sequence number is only used by the service layer. Metadata can also be used for OAM indications and instrumentation of DetNet data plane operation.

Generally speaking, more data plane metadata and related processing SHOULD be supported in the large-scale deterministic networks. By introducing IPv6 Extension Headers [RFC8200] and Segment Routing over IPv6 [RFC8986], native IPv6 data plane should be supported with the IPv6-sepcific enhancement. This section lists the data plane enhancement requirements based on but not limited to the technical requirements in Section 3, describing how to use IP and/or MPLS, and related OAM, to support a data plane method of flow identification and packet treatment over Layer 3.

4.1. Support Aggregated Flow Identification

Current IPv6 aggregated flow identification is generally based on 5 or 6 tuples, IP prefixes, or wildcards as indicated in [RFC8938]. However, in large-scale deterministic networks the number of individual flows is huge, and they may randomly join and leave the aggregated flow at each hop. Such behaviours lead to the difficulty in identifying aggregated flows by relying on the prefixes or wildcards.

In addition, the deterministic services may demand different deterministic QoS requirements according to different levels of application requirements. The flow identification with service-level aggregation should be supported. Flow identification is also used to quickly push a packet to a suitable queue. In a large-scale network, there are large amount of flows requiring deterministic latency service and normal forwarding service. Explicit flow identification makes it easier to quickly distinguish the DetNet flows without requiring the longest match rule on multiple tuples in IP data plane. Therefore, explicit aggregated flow identification SHOULD be supported.

4.2. Support Information used by Functions ensuring Deterministic Latency

According to Section 3.1, a large-scale network should support synchronized or asynchronized queuing mechanisms. Different queuing mechanisms require different information to be defined as the DetNet-specific metadata to help the functions of ensuring deterministic latency, including regulation, queue management.etc. For instance, the data plane needs to support the identification of cycle for cyclic queuing and forwarding or the latency related information for time based queuing in order to enable the applicability of the respective queuing and/or scheduling mechanisms in the large scale network.

When different queuing mechanisms are deployed at a network node, metadata used for each queuing mechanism should be provided at the same time. When multiple metadata carried in one packet, metadata should be self-described and extensible to tolerate variable number of metadata. Meanwhile, extra data will cause extra processing, referring to fixed or variable length lookups, total number of read/write access to the packet header.etc. So the data plane processing efficiency also needs to be considered when ensuring deterministic latency, but the specific method or solution is out of scope of this document.

This document does not specify what metadata and what format to be carried in data plane. The solution document should be specific enough on why and how the information carried as data plane meta data works in conjunction with the queuing or other functions and how it helps the large scale network deployment.

4.3. Support Redundancy Related Fields

Sequence number is the only metadata currently defined for redundancy feature of Detnet. MPLS data plane uses Detnet-over-MPLS label stack to carry it. At the same time, native IPv6 data plane should be able to carry this information too. If specific IP encapsulation or tunnel is in use, this meta data should be defined explicitly for that data plane.

4.4. Support Explicit Path Selection

Explicit route at the control plane and/or management is required so that the "best" path can be selected to meet the latency requirement for DetNet flows. At the data planes, MPLS label stack can be used for this purpose. IP data plane enhancement is required to support the explicit path selection based on IP source routing or SRv6.

5. Conclusion

This document specifies the technical requirements when ensuring the deterministic features in the large-scale networks, and the corresponding data plane enhancement requirements to support the them. Some of the proposed queuing mechanisms and trials are cited and the authors of the document think those proposals give reasonably sound insights to enhancement the current queuing mechanisms to meet the deterministic requirements of the large-scale networks.

6. Security Considerations

There are no IANA actions required by this document.

7. IANA Considerations

This section will be described later.

8. Acknowledgements

The authors would like to thank Lou Berger, Bala'zs Varga, Fan Yang, Tianran Zhou,Yaakov Stein for helpful suggestions. The authors also would like to thank Liang Geng, Peter Willis, Shunsuke Homma and Li Qiang for their previous works.

9. Contributors

The following people have substantially contributed to this document:

Zongpeng Du
China Mobile
EMail: duzongpeng@chinamobile.com

Lei Zhou
New H3C Technologies
Email: zhou.leih@h3c.com

10. Normative References

[Fast-Ethernet-MII-clock]
"Fast Ethernet MII clock".

[G.8262] International Telecommunication Union, "Timing characteristics of a synchronous equipment slave clock", ITU-T Recommendation G.8262, November 2018.

[G.8273] International Telecommunication Union, "Framework of phase and time clocks", ITU-T Recommendation G.8273, March 2018.

[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.du-detnet-layer3-low-latency]
Du, Z. and P. Liu, "Micro-burst Decreasing in Layer3 Network for Low-Latency Traffic", Work in Progress, Internet-Draft, draft-du-detnet-layer3-low-latency-05, 7 July 2022, <https://www.ietf.org/archive/id/draft-du-detnet-layer3-low-latency-05.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.geng-detnet-requirements-bounded-latency]
Geng, L., Willis, P., Homma, S., and L. Qiang, "Requirements of Layer 3 Deterministic Latency Service", Work in Progress, Internet-Draft, draft-geng-detnet-requirements-bounded-latency-03, 7 July 2019, <https://www.ietf.org/archive/id/draft-geng-detnet-requirements-bounded-latency-03.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>.

[IEEE802.1Qav]
IEEE, "IEEE Standard for Local and metropolitan area networks -- Virtual Bridged Local Area Networks - Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams", IEEE 802.1Qav-2009, DOI 10.1109/IEEESTD.2010.8684664, 5 January 2010, <https://doi.org/10.1109/IEEESTD.2010.8684664>.

[IEEE802.1Qbv]
IEEE, "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic", IEEE 802.1Qbv-2015, DOI 10.1109/IEEESTD.2016.8613095, 18 March 2016, <https://doi.org/10.1109/IEEESTD.2016.8613095>.

[IEEE802.1Qch]
IEEE, "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 29: Cyclic Queuing and Forwarding", IEEE 802.1Qch-2017, DOI 10.1109/IEEESTD.2017.7961303, 28 June 2017, <https://doi.org/10.1109/IEEESTD.2017.7961303>.

[IEEE802.1Qcr]
IEEE, "IEEE Standard for Local and Metropolitan Area Networks -- Bridges and Bridged Networks - Amendment 34: Asynchronous Traffic Shaping", IEEE 802.1Qcr-2020, DOI 10.1109/IEEESTD.2020.9253013, 6 November 2020, <https://doi.org/10.1109/IEEESTD.2020.9253013>.

[IEEE802.1TSN]
IEEE Standards Association, "IEEE 802.1 Time-Sensitive Networking Task Group", <https://www.ieee802.org/1/pages/tsn.html>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8578] Grossman, E., Ed., "Deterministic Networking Use Cases", RFC 8578, DOI 10.17487/RFC8578, May 2019, <https://www.rfc-editor.org/info/rfc8578>.

[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>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[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>.

Appendix A. Examples of Large-Scale Deterministic Network Trials

Some trials have been carried out to verify the concept of large-scale deterministic networks.

In order to verify the deterministic technology of large-scale networks, a trial of Deterministic IP on China Environment for Network Innovations (CENI), which is a network built for new network technology trial, was deployed. A network with a distance of 3,000 km over 13 hops was tested, and the jitter was controlled within 100us.

In order to verify the remote control on Deterministic IP, which required that the latency should be controlled within 4 ms and jitter should be controlled within 20 us. A trial cooperated with Baosteel spanned 600 km was deployed. Baosteel is a Chinese steel company and put forward this demand. Both of the first and second trials are based on a frequency synchronization solution. The mechanism details could be found in . [I-D.dang-queuing-with-multiple-cyclic-buffers][I-D.qiang-detnet-large-scale-detnet].

In order to realize multi flows synchronization on an inter-provincial network in an exhibition, Emergen proposed the requirement that two flows of video and virtual reality (VR) were sent from province A, and arrived at province B together, so people can see the synchronization of video collected by camera and the VR model. This requirement was proposed to facilitate the virtual industry product deployment. Due to time and other problems, it was realized by the edge network device for a relatively lower levels of service level agreement (SLA).

Teaming up with a smart factory operator, network operators, equipment companies, and universities, ETRI demonstrated an ultra-low latency, high-reliability 5G wired and wireless network-based remote industrial Internet of Things (IIoT) service by connecting a control center and a smart factory through three different operators' networks at a distance of 280 km. In this trail, it was demonstrated

that real-time remote smart manufacturing service is possible by making round-trip delay below 3 ms within a smart factory and below 10 ms between remote 5G industrial devices. In the future, the team plans to examine feasibility of large-scale deterministic networking by connecting smart factories in Gyeongsan, South Korea and Oulu, Finland.

These trials show that both operators and enterprise users begin to put forward requirements for the certainty of large-scale networks, but the implementation technologies are not exactly the same.

Authors' Addresses

Peng Liu
China Mobile
Beijing
100053
China
Email: liupengyjy@chinamobile.com

Yizhou Li
Huawei
Nanjing
210012
China
Email: liyizhou@huawei.com

Toerless Eckert
Futurewei Technologies USA
Santa Clara, 95014
United States of America
Email: tte@cs.fau.de

Quan Xiong
ZTE Corporation
Wuhan
430223
China
Email: xiong.quan@zte.com.cn

Jeong-dong Ryoo
ETRI
Daejeon
34129
South Korea
Email: ryoo@etri.re.kr

Shiyin Zhu
New H3C Technologies
Beijing
100094
China
Email: zhushiyin@h3c.com

Xuesong Geng
Huawei
Beijing
100095
China
Email: gengxuesong@huawei.com

DetNet Working Group
Internet-Draft
Intended status: Informational
Expires: 27 April 2023

J. Joung
Sangmyung University
J. Ryoo
T. Cheung
ETRI
Y. Li
Huawei
P. Liu
China Mobile

24 October 2022

Asynchronous Deterministic Networking Framework for Large-Scale Networks
draft-joung-detnet-asynch-detnet-framework-01

Abstract

This document describes an overall framework of Asynchronous Deterministic Networking (ADN) for large-scale networks. It specifies the functional architecture and requirements for providing latency and jitter bounds to high priority traffic, without strict time-synchronization of network nodes.

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 27 April 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Deterministic Networking (DetNet) provides a capability to carry specified unicast or multicast data flows for real-time applications with extremely low data loss rates and bounded latency within a network domain. The architecture of DetNet is defined in RFC 8655 [RFC8655], and the overall framework for DetNet data plane is provided in RFC 8938 [RFC8938]. Various documents on DetNet IP (Internet Protocol) and MPLS (Multi-Protocol Label Switching) data

planes and their interworking with Time-Sensitive Networking (TSN) have been standardized. Technical elements necessary to extend DetNet to a large-scale network spanning multiple administrative domains are identified in [I-D.liu-detnet-large-scale-requirements].

This document considers the problem of guaranteeing both latency upper bounds and jitter upper bounds in large-scale networks with any type of topology, with random dynamic input traffic. The jitter is defined as the latency difference between two packets within a flow, not a difference from a clock signal or from an average latency, as is summarized in RFC 3393 [RFC3393].

In large-scale networks, the end-nodes join and leave, and a large number of flows are dynamically generated and terminated. Achieving satisfactory deterministic performance in such environments would be challenging. The current Internet, which has adopted the DiffServ architecture, has the problem of the burst accumulation and the cyclic dependency, which is mainly due to FIFO queuing and strict priority scheduling. Cyclic dependency is defined as a situation wherein the graph of interference between flow paths has cycles [THOMAS]. The existence of such cyclic dependencies makes the proof of determinism a much more challenging issue and can lead to system instability, that is, unbounded delays [ANDREWS][BOUILLARD]. The Internet architecture does not have an explicit solution for the jitter bound as well. Solving the problem of latency and jitter as a joint optimization problem would be even more difficult.

The basic philosophy behind the framework proposed in this document is to minimize the latency bounds first by taking advantage of the work conserving schedulers with regulators or stateless fair queuing schedulers, and then minimize the jitter bounds by adjusting the packet inter-departure times to reproduce the inter-arrival times, at the boundary of a network. We argue that this is simpler than trying to minimize the latency and the jitter at the same time. The direct benefit of such simplicity is its scalability.

For the first problem of guaranteeing latency bound alone, the IEEE asynchronous traffic shaping (ATS) [IEEE802.1Qcr], the flow-aggregate interleaved regulators (FAIR) [FAIR][Y.3113] frameworks, the port-based flow aggregate regulators (PFAR) [ADN], and the work-conserving stateless core fair queuing (C-SCORE) are proposed as solutions. The key component of the ATS and the FAIR frameworks is the interleaved regulator (IR)), which is described in [I-D.ietf-detnet-bounded-latency]. The IR has a single queue for all flows of the same class from the same input port. The head of the queue (HOQ) is examined if the packet is eligible to exit the regulator. To decide whether it is eligible, the IR must maintain the individual flow states. The key component of the PFAR framework

is the regulators for flow aggregates (FA) per port per class, which regulates the FA based on the sum of average rates and the sum of maximum bursts of the flows that belong to the FA. In the meantime, the key component of the C-SCORE is the packet state that is carried as meta-data. The C-SCORE does not need to maintain flow states at core nodes, yet it is one of the fair queuing schedulers. The service order of the packet is directly inferred from the packet state. It can be implemented based on per-input port FIFO queues. The meta-data to be carried in the packet header is simple and can be updated during the stay in the queue or before joining the queue.

For the second problem of guaranteeing jitter bound, it is necessary to assume that the first problem is solved, that is, the network guarantees latency bounds. Furthermore, the network is required to specify the value of the latency bound for a flow. The end systems at the network boundary, or at the source and destination nodes, then can adjust the inter-departure times of packets, such that they are similar to their inter-arrival times. In order to identify the inter-arrival times at the destination node, or at the network edge near the destination, the packets are required to specify their arrival times, according to the clock at the source, or the network edge near the source. The clocks are not required to be time-synchronized with any other clocks in a network. In order to avoid a possible error due to a clock drift between a source and a destination, they are recommended to be frequency-synchronized.

In this document, strict time-synchronization among network nodes is avoided. It is not easily achievable, especially over a large area network or across multiple DetNet domains. Asynchronous solutions suggested in this document can provide satisfactory latency bounds with careful design without complex pre-computation, configuration, and hardware support usually necessary for time synchronization.

2. Terminology

2.1. Terms Used in This Document

2.2. Abbreviations

3. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. Framework for Latency Guarantee

4.1. Problem Statement

In Section 4, we assume there are only two classes of traffic. The high priority traffic requires latency upper bound guarantee. All the other traffic is considered to be the low priority traffic, and be completely preempted by the high priority traffic. High priority (HP) traffic is our only focus.

It is well understood that the necessary conditions for a flow to have a bounded latency inside a network, are that;

* a flow entering a network conforms to a prescribed traffic specification (TSpec), including the arrival rate and the maximum burst size, and

* all the network nodes serve the flow with a service rate which are greater than or equal to the arrival rate.

These conditions make the resource reservation and the admission control mandatory. These two functions are considered given and out of scope of this document.

Here, the notion of arrival and service rates represent sustainable or average values. A short-term discrepancy between these two rates contributes to the burst size increment, which can be accumulated as the flow passes through the downstream nodes. This results in an increase in the latency bound. Therefore, the value of accumulated burst size is a critical performance metric.

The queuing and scheduling of a flow plays a key role in deciding the accumulated burst size. Ideally, the flows can be queued in separate queues and the queues are scheduled according to the flow rates. In this case a flow can be considered protected. With practical fair schedulers, such as the Deficit Round Robin (DRR), a protected flow still can be affected by the other flows as much as their maximum packet lengths.

If we adopt a separate queue per flow at an output port, and assume identical flows from all the input ports, then the maximum burst size of a flow out of the port, Bout, is given as the following:

$$Bout < Bin + (n-1)L*r/C,$$

where Bout is the outgoing flow's maximum burst size, Bin is the incoming flow's maximum burst size, n is the number of the flows, L is the maximum packet size, r is the average rate of the flow, and C is the link capacity. This approach was taken in the integrated services (IntServ) framework [RFC2212].

The separate queues in the aforementioned case can be too many to be handled in real time, especially at the core of large-scale networks. The common practice therefore is to put all the HP flows in a single queue, and serve them with higher priority than best effort traffic. It is also well known that a proper scheduling scheme, such as the strict priority (SP) scheduling can guarantee service rates larger than the arrival rates, therefore the latency can still be guaranteed. With such a single aggregate queue the flows are not considered protected, however. In this case a flow's burst size in a node can be increased proportionally to the sum of maximum burst sizes of the other flows in the queue. That is,

$$Bout < Bin + (n-1)Bin*r/C.$$

The second product term on the right-hand side represents the amount of increased maximum burst. It is dominated by the term (n-1)Bin, which is the maximum total burst from the other flows.

Moreover, this increased burst affects the other flows' burst size at the next node, and this feedforward can continue indefinitely where a cycle is formed in a network. This phenomenon is called a cyclic dependency of a network. It is argued that the burst accumulation can explode into infinity, therefore the latency is no longer guaranteed.

As such, a flow is required to be protected to a certain level, from the other flows' bursts, such that its burst accumulations are kept within a necessary value. By doing so, the other flows are also protected. The regulators or the fair queuing schedulers are proposed as solutions for such protection in this document. They can decrease the accumulated burst into a desirable level and can protect flows from others. In case of the regulators, however, if the regulation needs a separate queue per flow, then the scalability would be harmed just like the ideal IntServ case. In this document the IR or the regulations on flow aggregates are proposed.

The key requirement for latency guarantee is therefore to have scalability and a certain level of flow protection.

4.2. Asynchronous Traffic Shaping (ATS)

The first solution in this document for latency guarantee is the IEEE TSN TG's ATS technology. Essentially it is a combined effort of the flow aggregation per node per input/output ports pair per class, and the interleaved regulator per flow aggregate (FA). The IR examines the HOQ, identifies the flow the packet belongs to, and transfers the packet only when it is eligible according to the initial TSpec of the flow. This solution can have only one queue per FA, but suffers from having to maintain each individual flow state. The detailed description on ATS can be found in [IEEE802.1Qcr].

4.3. Flow Aggregate Interleaved Regulators (FAIR)

4.3.1. Overview of the FAIR

In the FAIR framework, the network can be divided into several aggregation domains (ADs). HP flows of the same path within an AD are aggregated into an FA. IRs per FA are implemented at the boundaries of the ADs. An AD can consist of arbitrary number of nodes. The FA can be further subdivided based on the flow requirements and characteristics. For example, only video flows of the same path are aggregated into a single FA.

Figure 1 shows an example architecture of the FAIR framework. The IRs at the AD boundaries suppress the burst accumulations across the ADs with the latency upper bounds intact as they do in IEEE TSN ATS, if the incoming flows are all properly regulated, and the AD guarantees the FIFO property to all the packets in the FA [LEBOUDEC]. It is sufficient to put every FA into a single FIFO queue in a node, in order to maintain the FIFO property within an AD. However, in this case, if cycles are formed, the burst accumulations inside an AD can be accumulated indefinitely. If the topology does not include a cycle and the latency bound requirement is not stringent, then the FIFO queue and the SP scheduler would be allowable. Otherwise, the FAs are recommended to be treated with separated queues and fair-queuing schedulers for flow protection.

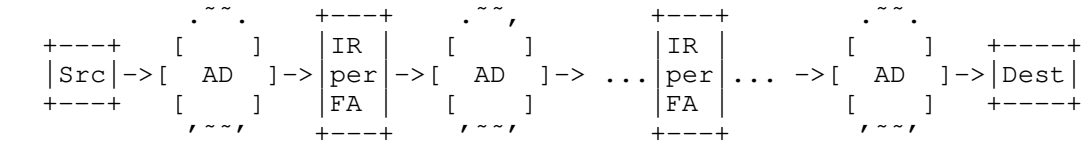


Figure 1: FAIR Framework

4.3.2. The performance of the FAIR

The FAIR guarantees an end-to-end delay bound with reduced complexity compared to the traditional flow-based approach. Numerical analysis shows that, with a careful selection of AD size, the FAIR with DRR schedulers yields smaller latency bounds than both the IntServ and the ATS [FAIR].

The ATS can be considered as a special case of the FAIR with the FIFO schedulers, where all the ADs encompass only a single hop. The IntServ can also be considered as an extreme case of the FAIR with fair schedulers and queues per FA, with an AD corresponding to an entire network; therefore, regulators are unnecessary.

4.4. Port-based Flow Aggregate Regulators (PFAR)

The IR in the ATS and the FAIR suffers from two major complex tasks; the flow state maintenance and the HOQ lookup to determine the flow to which the packet belongs. Both tasks involve real-time packet processing and queue management. As the number of flows increases, the IR operation may become burdensome as much as the per- flow regulators. Without maintaining individual flow states, however, the flows can be protected to a certain level, as is described in this section.

The ATS and FAIR mitigates the burst increment by placing IRs behind a FIFO system. For example, consider an ATS node with a single queue at an output port for HP traffic. The IR assigned for an input port forms a single queue for the flows from the same input port. Further consider the set of incoming flows from the same input port of the ATS node. Let us call this set of flows the incoming flow aggregate (FAin). If we assume identical FAins from all the input ports, then the maximum burst size of an arbitrary set of flows out of the port, Bout, is given as the following:

$$Bout < Bin + (p-1)B*r/C,$$

where Bin is the sum of maximum burst sizes of the flows within the FAin, B is the sum of initial maximum burst sizes of the flows within the FAin, and p is the number of the ports in the node.

The port-based FA (PFA) is defined as a set of HP flows in the same class, which share the input and output ports in a relay node, such as a switch or router. The only aggregation criteria for a PFA are the ports and the class. The port-based flow aggregate regulators (PFAR) framework puts a regulator for each PFA in an output port module, just before the class-based queuing/scheduling system of the output port module. The PFAR framework sees a PFA as a single flow

with the "PFA-Tspec", {the sum of the maximum initial bursts; and the sum of the initial arrival rates} of the flows that are the elements of the PFA; and regulates the PFA to meet its PFA-Tspec.

The PFARs can be placed at the output port of a node before the output SP scheduler. The architecture is similar to that suggested in the IEEE ATS, except that in the ATS, the IRs are placed instead of the PFARs.

The burst increment of an FA in the PFAR architecture is identical to that in the ATS, which is given as;

$$Bout < Bin + (p-1)B*r/C,$$

where B is again the initial maximum burst size of FAs. However, the regulators in PFAR does introduce additional latency, which is given as

$$D < (Bin - B)/r,$$

where D is the latency within the regulator.

Note that Bout is a function of (n-1)B, not (n-1)Bin; in other words, the burst size out of a node is affected only by the initial burst sizes of the other FAs from different input ports of the node. This property makes the D or Bout do not increase exponentially even in the existence of cyclic dependencies.

With the PFAR, the HOQ flow identification process is unnecessary, and only the PFAs' states, instead of individual flows' states, must be maintained at a node. In this respect, the complexity of process of PFAR is reduced compared to IR of the ATS or the FAIR.

In a recent study [ADN], it was also shown, through a numerical analysis with symmetrical networks with cycles, that PFAR, when implemented at every node, can achieve comparable latency bounds to the IEEE ATS technique.

The ATS, the FAIR, and the PFAR frameworks maintain regulators per FA. The FAs in these frameworks are composed of the flows sharing the same ingress/egress ports of an AD. The ADs can encompass a single hop or multiple hops. The regulators can be the IR or the aggregate regulator. There can be other combinations of AD and regulator type, which could be further investigated and compared to the frameworks introduced in this document.

4.5. Work-conserving stateless core fair queuing (C-SCORE)

The generalized processor sharing (GPS) [PAREKH], the weighted fair queuing (WFQ), the virtual clock (VC), and similar other schedulers utilize the concept of finish time (FT) that is the service order assigned to a packet. The packet with the minimum FT in a buffer is served first. We will call these works collectively as the fair queuing (FQ).

As an example, the VC scheduler [ZHANG] defines the FT to be

$$F(p) = \max\{F(p-1), A(p)\} + L(p)/r, \quad (1)$$

where (p-1) and p are consecutive packets of the flow under observation, A(p)is the arrival time of p, L(p) is the length of p, and r is the flow service rate. The flow index is omitted.

The key idea of the FQ is to calculate the service finish times of packets in an imaginary ideal fluid service model and use them as the service order in the real packet-based scheduler.

While having the excellent flow protection property, the FQ needs to maintain the flow state, F(p-1). For every arriving packet, the flow it belongs to has to be identified and its previous packet's FT should be extracted. As the packet departs, the flow state, F(p), has to be updated as well.

We consider a framework for constructing FTs for packets at core nodes without flow states. In a core node, the following conditions on FTs have to be met.

C1) It has to keep the 'fair distance' of consecutive packets of a flow. That is; Fh(p) >= Fh(p-1) + L(p)/r, where Fh(p) is the F(p) at node h.

C2) The order of FTs and the actual service order, within a flow, have to be kept. That is; Fh(p) > Fh(p-1) and Ch(p) > Ch(p-1), where Ch(p) is the actual service completion time of packet p at node h.

C3) The time lapse at each hop has to be reflected. That is; Fh(p) >= F(h-1)(p), where F(h-1)(p) is the FT of p at the node h-1, the upstream node of h.

C4) The FTs of a flow have to be aligned to the packet arrival times. That is; L(p)/r <= Fh(p)- Ah(p) < Delta.

Delta can be any finite positive value [STILIADIS]. In other words, the Fh(p) should be larger than Ah(p)+L(p)/r, as in (1), yet still should grow at the same rate as Ah(p).

In essence, (1) has to be approximated in core nodes. There can be many possible solutions to meet these conditions. We propose a generic framework for constructing FTs in core nodes, without flow state, in the following.

We denote a 'node' to be an output port of a relay node.

Requirement 1: In the entrance node, it is required to obtain the FTs with (1). That is to obtain F0(p) as in the VC, where 0 denotes the entrance node of the flow under observation.

$$F0(p) = \max\{F0(p-1), A0(p)\}+L(p)/r.$$

Note that F0(p) keeps the fair distances from the FTs of consecutive packets of the flow.

Requirement 2: It is required to increase the FT of a packet by an amount that depends on the node and the packet, dh(p), in a core node h.

$$Fh(p) = F(h-1)(p) + d(h-1)(p).$$

Requirement 3: It is required that dh(p) is a non-decreasing function of p, within a node busy period.

Definition 1: A node busy period is a maximal interval between consecutive node idle periods. During a node idle period, the node has no packet to send.

By Requirements 1, 2, and 3; Conditions 1), 2), and 3) are met.

Requirement 4: It is required that Ah(p)+dh(p) >= A(h+1)(p).

One example of dh(p) is a measured maximum latency of a packet in the node h up until the current packet p, since the start of a node busy period. Let us denote this local maximum latency with uh(p). It may be reset to an initial value during a node idle period. An example of the initial value of uh(p) is the propagation delay from node h to (h+1). By letting dh(p)=uh(p), Requirement 4 is satisfied.

dh(p) may not be a function of p, and dependent only on the node. Then it could be denoted as dh.

One example of dh is letting dh = Uh, where Uh is the latency upper bound in node h for any p. Uh can be a theoretical one, or be obtained by long-term measurements. By letting dh(p)=Uh, Requirement 4 is satisfied.

If Requirement 4 is satisfied then it can be guaranteed Fh(p) >= Ah(p)+L(p)/r, for all h>=0, and it can be proven that Condition 4) is met.

In a core node, the service order of packets from the same input port can be preserved. That is, if Ah(p)> Ah(p') then Ch(p)>Ch(p') for packets p and p' that travel together the nodes (h-1) and h. By preserving the service order of packets from the same input port, using per-input port FIFO queues is possible. An example implementation would be as the follows: The output port module is composed of per-input port FIFO queues. As a packet enters the FIFO queue according to its input port, it should join the queue at the tail and be marked with its FT. The scheduler will examine the smallest FT among the packets at the HoQ of the FIFO queues.

Note that Ah(p)> Ah(p') does not guarantee Fh(p)>Fh(p') when p and p' belong to different flows. For example, p' may have a smaller FT but arrive later while p is in service. However, it is proven that this service completion time discrepancy, C0(p)-F0(p), between real packet system and ideal fluid system is bounded by Lmax/C [PAREKH], where Lmax is the maximum packet length over all the flows, and C is the link capacity.

The meta-data to carry in a packet are Fh(p) and dh(p). These are dynamic and thus need to be updated at every hop. Note that if dh(p) = dh then it can also be signaled out-of-band between the neighboring nodes. Fh(p) can be obtained by a simple summation of two meta-data, and updated during the time interval between the packet arrival and its reaching HoQ of the FIFO queue.

The proposed FT construction framework has advantages of simple FIFO-based implementation and simple meta-data management. We call this solution the work conserving stateless core fair queuing (C-SCORE), which can be compared to the existing non-work conserving scheme [STOICA].

5. Framework for Jitter Guarantee

5.1. Problem statement

The problem of guaranteeing jitter bounds in arbitrarily sized networks with any type of topology with random dynamic input traffic is considered.

There are several possible solutions to guarantee jitter bounds in packet networks, such as IEEE TSN's cyclic queuing and forwarding (CQF) [IEEE802.1Qch], its asynchronous variations [I-D.yizhou-detnet-ipv6-options-for-cqf-variant], and the latency-based forwarding (LBF) [LBF].

The CQF requires time-synchronization across every node in the network including the source. It is not scalable to a large network with significant propagation delays between the nodes. The asynchronous CQFs are scalable, but they may not satisfy applications' jitter requirements. This is because their jitter bounds cannot be controlled as desired, but are only determined by the cycle time, which should be large enough to accommodate all the traffic to be forwarded.

The systems with slotted operations such as the CQF and its variations turn the problem of packet scheduling into the problem of scheduling flows to fit into slots. The difficulty of such a slot scheduling is a significant drawback in large scale dynamic networks with irregular traffic generations and various propagation delays.

The LBF is a framework of the forwarding action decision based on the flow and packet status, such as the delay budget left for a packet in a node. The LBF does not specify the actions to take according to the status. It suggests a packet slow down or speedup by changing the service order, by pushing packets into any desirable position of a first out queue, as a possible action to take. In essence, by having latency budget information of every packet, the LBF is expected to maintain the latency and jitter within desired bounds. The processing latency required in LBF includes times 1) to lookup the latency budget information on every packet header, 2) to decide the queue position of the packet, 3) to modify the queue linked list, and 4) to update the budget information on the packet upon transmission. This processing latency, however, can affect the scalability especially in high speed core networks.

The ATS, the FAIR, and the PFAR utilize the regulation function to proactively prevent the possible burst accumulation in the downstream nodes. It is not clear whether the LBF can take such preventive action. If so the LBF can also act as a regulator and yield a similar latency bound.

5.2. Buffered network (BN)

The BN framework in this document for jitter bound guarantee is composed of

* a network that guarantees latency upper bounds;

* a timestamper for packets with a clock that is not necessarily synchronized with the other nodes, which resides in between, including the source and the network ingress interface; and

* a buffer that can hold the packets for a predetermined interval, which resides in between, including the destination and the network egress interface.

Figure 2 depicts the overall architecture of the BN framework for jitter-bound guarantees [BN]. Only a single flow is depicted between the source and destination in Figure 2. The arrival (an), departure (bn), and buffer-out (cn) times of the nth packet of a flow are denoted. The end-to-end (E2E) latency and the E2E buffered latency are defined as (bn-an) and (cn-an), respectively.

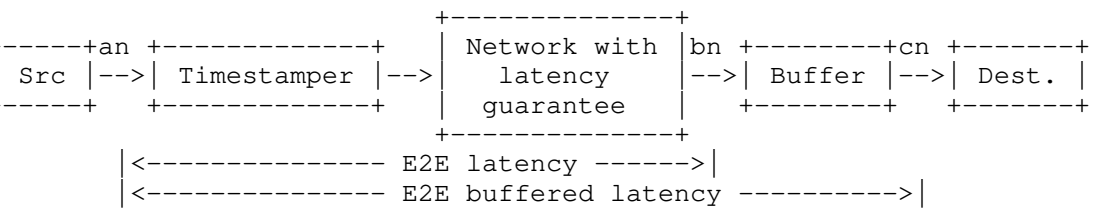


Figure 2: Buffered Network (BN) Framework for Jitter Guarantee

The buffer supports as many as the number of the flows destined for the destination. The destination shown in Figure 2 can be an end station or another deterministic network. The buffer holds packets in a flow according to predefined intervals. The decision of the buffering intervals involves the time-stamp value within each packet.

The network in between the time-stamper and the buffer can be of arbitrarily sized network. The input traffic can be dynamic. It is required that the network be able to guarantee and identify the E2E latency upper bounds of the flows. The network is also required to let the buffer be aware of the E2E latency upper bounds of the flows it has to process. It is recommended that the E2E latency lower bound information is provided by the network as well. The lower bound may be contributed from the transmission and propagation delays within the network.

The time-stamper marks on the packets their arrival times. The time-stamping function can use the real-time transport protocol (RTP) over the user datagram protocol (UDP) or the transmission control protocol (TCP). Either the source or network ingress interface can stamp the packet. In the case where the source stamps, the timestamp value is the packet departure time from the source, which is only a propagation time away from the packet arrival time to the network.

The source and destination do not need to share a synchronized clock. All we need to know is the differences between the time stamps, that is, the information about the inter-arrival times.

5.3. Properties of the BN

Let the arrival time of the nth packet of a flow be an. Similarly, let bn be the departure time from the network of the nth packet. Then, a1 and b1 are the arrival and departure times of the first packet of the flow, respectively. The first packet of a flow is defined as the first packet generated by the source, among all the packets that belong to the flow. Further, let cn be the buffer-out time of the nth packet of the flow. Let us define m as the jitter control parameter, which will be described later in detail.

Since buffers can be without cut-through capability, the processing delay within a buffer has to be taken in account. Let gn be the processing delay within the buffer of the nth packet of the flow. The gn includes the time to look up the timestamp and to store/ forward the packet. However, it does not include an intentional buffer-holding interval. By definition, cn - bn >= gn. Let max_n(gn)=g, the maximum processing delay for the flow in the buffer. It is assumed that a buffer can identify the value of g. Let U and W be the latency upper and lower bounds guaranteed to the flow by the network. Let m be the jitter control parameter, W+g <= m.

The rules for the buffer-holding interval decision are given as follows:

* c1=(b1+m-W),

* cn=max{(bn+g), (c1+an-a1)}, for n > 1.

The second rule governing the cn states that a packet should be held in the buffer to make its inter-buffer-out time, (cn-c1), equal to the inter-arrival time, (an-a1). However, when its departure from the network is too late, the inter-buffer-out time should be larger than the inter-arrival time, then hold the packet as much as the maximum processing delay in the buffer, that is, cn=bn+g. The buffer does not need to know the exact values of an or a1. It is sufficient to determine the difference between these values, which can be easily obtained by subtracting the timestamp values of the two packets.

The following theorems holds [ADN].

Theorem 1 (Upper bound of E2E buffered latency). The latency from the packet arrival to the buffer-out times (cn-an), is upper bounded by (U-W+m).

Theorem 2 (Lower bound of E2E buffered latency). The latency from the packet arrival to the buffer-out times (cn-an), is lower bounded by m.

Theorem 3 (Upper bound of jitter). The jitter is upper bounded by max{0, (U+g-m)}.

By setting m=(U+g), we can achieve zero jitter. In this case, the E2E buffered latency bound becomes (2U+g-W), which is roughly twice the E2E latency bound. In contrast, if we set m to its minimum possible value W+g, then the jitter bound becomes (U-W), which is roughly equal to U, while the E2E buffered latency bound becomes U, which is the same as the E2E latency bound.

The parameter m directly controls the holding interval of the first packet. It plays a critical role in determining the jitter and the buffered latency upper bounds of a flow in the BN framework. The larger the m, the smaller the jitter bound, and the larger the latency bound. With a sufficiently large m, we can guarantee zero jitter, at the cost of an increased latency bound.

5.4. Frequency synchronization between the source and the buffer

Clock drift refers to phenomena wherein a clock does not run at exactly the same rate as a reference clock. If we do not frequency-synchronize the clocks of different nodes in a network, clock drift is unavoidable. Consequently, jitter occurs owing to the clock frequency difference or clock drift between the source (timestamper) and the buffer. Therefore, it is recommended to frequency-synchronize the source (timestamper) and the buffer.

5.5. Omission of the timestamper

For isochronous traffic whose inter-arrival times are well-known fixed values, and the network can preserve the FIFO property for such traffic, then the timestampers can be omitted.

Otherwise the FIFO property cannot be guaranteed, then a sequence number field in the packet header would be enough to replace the timestamper.

5.6. Mitigation of the increased E2E buffered latency

The increased E2E buffered latency bound by the proposed framework, from U to almost 2U, can be mitigated by one of the added functionalities given as follows.

1) First, one can measure the E2E latency of a flow's first packet exactly, and buffer it to make its E2E buffered latency be U. Then, by following the rules given in Section 5.3, every subsequent packet will experience the same E2E buffered latency, which is U, with zero jitter. An example of the exact latency measurement may be performed by time-synchronization between the source (timestamper) and the buffer. However, how to measure the latency is for further investigation.

2) Second, one can expedite the first packet's service with a special treatment, to make its latency lower, compared to the other packets of the flow. If we can make the first packet's latency to be a small value d, then every packet will experience the same buffered latency d+U, with zero jitter. Considering that the E2E latency bound is calculated from the worst case in which rare events occur simultaneously, however, the first packet's latency is likely to be far less than what the bound suggests. Therefore, the special treatment to the first packet may be ineffective in real implementations.

5.7. Multi-sources single-destination flows' jitter control

The BN framework can also be used for jitter control among multiple sources' flows having a single destination. When a session is composed of more than one sources, physically or virtually separated, the buffer at the boundary can mitigate the latency variations of packets from different sources due to different routes or network treatments. Such a scenario may arise in cases such as

1) that a central unit controls multiple devices for a coordinated execution in smart factories, or

2) multi-user conferencing applications, in which multiple devices/users physically separated can have a difficulty in real-time interactions.

The sources, or the ingress boundary nodes of the network, need to be synchronized with each other in order for the time-stamps from separated sources to be able to identify the absolute arrival times.

6. IANA Considerations

There are no IANA actions required by this document.

7. Security Considerations

This section will be described later.

8. Acknowledgements

9. Contributor

10. References

10.1. Normative References

[I-D.ietf-detnet-bounded-latency]
Finn, N., Boudec, J. L., 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.liu-detnet-large-scale-requirements]
Liu, P., Li, Y., Eckert, T., Xiong, Q., Ryoo, J., Zhu, S., and X. Geng, "Requirements for Large-Scale Deterministic Networks", Work in Progress, Internet-Draft, draft-liu-detnet-large-scale-requirements-05, 20 October 2022, <https://datatracker.ietf.org/api/v1/doc/document/draft-liu-detnet-large-scale-requirements/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

10.2. Informative References

[ADN] Joung, J., Kwon, J., Ryoo, J., and T. Cheung, "Asynchronous Deterministic Network Based on the DiffServ Architecture", IEEE Access, vol. 10, pp. 15068-15083, doi:10.1109/ACCESS.2022.3146398, 2022.

[ANDREWS] Andrews, M., "Instability of FIFO in the permanent sessions model at arbitrarily small network loads", ACM Trans. Algorithms, vol. 5, no. 3, pp. 1-29, doi: 10.1145/1541885.1541894, July 2009.

[BN] Joung, J. and J. Kwon, "Zero jitter for deterministic networks without time-synchronization", IEEE Access, vol. 9, pp. 49398-49414, doi:10.1109/ACCESS.2021.3068515, 2021.

[BOUILLARD] Bouillard, A., Boyer, M., and E. Le Corronc, "Deterministic network calculus: From theory to practical implementation", in Networks and Telecommunications. Hoboken, NJ, USA: Wiley, doi: 10.1002/9781119440284, 2018.

[FAIR] Joung, J., "Framework for delay guarantee in multi-domain networks based on interleaved regulators", Electronics, vol. 9, no. 3, p. 436, doi:10.3390/electronics9030436, March 2020.

[I-D.yizhou-detnet-ipv6-options-for-cqf-variant] Li, Y., Ren, S., Li, G., Yang, F., Ryoo, J., and P. Liu, "IPv6 Options for Cyclic Queuing and Forwarding Variants", Work in Progress, Internet-Draft, draft-yizhou-detnet-ipv6-options-for-cqf-variant-00, 19 June 2022, <https://www.ietf.org/archive/id/draft-yizhou-detnet-ipv6-options-for-cqf-variant-00.txt>.

[IEEE802.1Qch] IEEE, "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 29: Cyclic Queuing and Forwarding", IEEE 802.1Qch-2017, DOI 10.1109/IEEESTD.2017.7961303, 28 June 2017, <https://doi.org/10.1109/IEEESTD.2017.7961303>.

[IEEE802.1Qcr] IEEE, "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks - Amendment 34: Asynchronous Traffic Shaping", IEEE 802.1Qcr-2020, DOI 10.1109/IEEESTD.2020.9253013, 6 November 2020, <https://doi.org/10.1109/IEEESTD.2020.9253013>.

[LBF] Clenm, A. and T. Eckert, "High-precision latency forwarding over packet-programmable networks", NOMS 2020 - IEEE/IFIP Network Operations and Management Symposium, April 2020.

[LEBOUDEC] Le Boudec, J., "A theory of traffic regulators for deterministic networks with application to interleaved regulators", IEEE/ACM Trans. Networking, vol. 26, no. 6, pp. 2721-2733, doi:10.1109/TNET.2018.2875191, December 2019.

[PAREKH] Parekh, A. and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case", IEEE/ACM Trans. Networking, vol. 1, no. 3, pp. 344-357, June 1993.

[RFC2212] Shenker, S., Partridge, C., and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, DOI 10.17487/RFC2212, September 1997, <https://www.rfc-editor.org/info/rfc2212>.

[RFC3393] Demichelis, C. and P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, DOI 10.17487/RFC3393, November 2002, <https://www.rfc-editor.org/info/rfc3393>.

[STILIADIS] Stiliadis, D. and A. Anujan, "Rate-proportional servers: A design methodology for fair queueing algorithms", IEEE/ACM Trans. Networking, vol. 6, no. 2, pp. 164-174, 1998.

[STOICA] Stoica, I. and H. Zhang, "Providing guaranteed services without per flow management", ACM SIGCOMM Computer Communication Review, vol. 29, no. 4, pp. 81-94, 1999.

[THOMAS] Thomas, L., Le Boudec, J., and A. Mifdaoui, "On cyclic dependencies and regulators in time-sensitive networks", in Proc. IEEE Real-Time Syst. Symp. (RTSS), York, U.K., pp. 299-311, December 2019.

[Y.3113] International Telecommunication Union, "Framework for Latency Guarantee in Large Scale Networks Including IMT-2020 Network", ITU-T Recommendation Y.3113, February 2021.

[ZHANG] Zhang, L., "Virtual clock: A new traffic control algorithm for packet switching networks", in Proc. ACM symposium on Communications architectures & protocols, pp. 19-29, 1990.

Authors' Addresses

Jinoo Joung
Sangmyung University
Email: jjoung@smu.ac.kr

Jeong-dong Ryoo
ETRI
Email: ryoo@etri.re.kr

Taesik Cheung
ETRI
Email: cts@etri.re.kr

Yizhou Li
Huawei
Email: liyizhou@huawei.com

Peng Liu
China Mobile
Email: liupengyjy@chinamobile.com

Network
Internet-Draft
Intended status: Standards Track
Expires: 25 April 2023

Shaofu. Peng
Bin. Tan
ZTE Corporation
Peng. Liu
China Mobile

22 October 2022

Deadline Based Deterministic Forwarding
draft-peng-detnet-deadline-based-forwarding-03

Abstract

This document describes a deterministic forwarding mechanism based on deadline. The mechanism enhances strict priority scheduling algorithm with dynamically adjusting the priority of the queue according to its deadline attribute.

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 25 April 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

[RFC8655] describes the architecture of deterministic network and defines the QoS goals of deterministic forwarding: Minimum and maximum end-to-end latency from source to destination, timely delivery, and bounded jitter (packet delay variation); packet loss ratio under various assumptions as to the operational states of the nodes and links; an upper bound on out-of-order packet delivery. In order to achieve these goals, deterministic networks use resource reservation, explicit routing, service protection and other means. Resource reservation refers to the occupation of resources by service traffic, exclusive or shared in a certain proportion, such as dedicated physical link, link bandwidth, queue resources, etc; Explicit routing means that the transmission path of traffic flow in the network needs to be selected in advance to ensure the stability of the route and does not change with the real-time change of network topology, and based on this, the upper bound of end-to-end delay and delay jitter can be accurately calculated; Service protection refers to sending multiple service flows along multiple disjoint paths at the same time to reduce the packet loss rate. In general, a deterministic path is a strictly explicit path calculated by a

centralized controller, and resources are reserved on the nodes along the path to meet the SLA requirements of deterministic services.

[I-D.stein-srtsn] describes that the controller calculates the local deadline time of each node for the traffic to be transmitted in advance, which is a absolute system time, forms a stack of these local deadline time, and then carries them in the forwarded data packets. Each node forwards the packets according to its own local deadline. [I-D.stein-srtsn] suggests that FIFO queue can not be used to realize this function, because the packets stored in the queue are always first in first out and a special data structure is recommoned. The packets in this data structure will be automatically sorted with the order from emergency to non emergency according to the deadline of the packets. However, it may be difficult to implement this structure in hardware, and especially for a large network it may be challenge to synchronize time.

Considering that the link propagation delay is generally a fixed value, and we focus on the residence time of the packets inside the node, an alternate approach is to make the deadline eliminate the interference of link propagation delay and avoids relying on time synchronization between nodes.

This document desrbies an alternate packets scheduling scheme based on EDF (earliest-deadline-firs) and used for wide area network. It suggests to only use a single deadline time to control the packets scheduling of all nodes along the path. The single deadline time is an offset time, which is based on the time when the packet enters the node and represents the maximum time allowed for the packet to stay inside the node. However, if each node has obvious differences in the capability of packets forwarding and scheduling, more offset-time may be needed.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Deadline Queue

For nodes in the network, some queues with count-down time (also termed as CT) are introduced and maintained for each outgoing port. These queues are called deadline queues and participate in priority based scheduling. Deadline queues have the following characteristics:

* The CT of each deadline queue will decrease with the passage of time. When it decreases to 0, the scheduling priority of the queue will be set to the highest, and the scheduling opportunity will be obtained immediately (note that there may be interference delay caused by a packet being sent by a low priority queue). It will prohibit receiving new packets, in which the buffered packets will be sent to the outgoing port immediately, and the maximum duration allowed to send packets is the preset authorization time (also termed as AT), e.g, 10us, 20us, etc. In principle, all packets buffered in the queue shall be sent within this authorization time. If the queue is sent to empty and the authorization time is still free, other queues with lower priority can be scheduled during this authorization time.

* The scheduling engine can initiate a cycle timer to decrement the CT of all deadline queues, that is, whenever the timer expires, the CT values of all queues will be subtracted from the timer interval (also termed as TI). Note that TI must be less than or equal to the AT, with AT = N * TI, where the natural number N >= 1.

* For a deadline queue whose CT has been reduced to 0, after a new round of authorization time, the CT will return to the maximum initial value (also termed as MAX_CT), allow receiving new packets, and continue to enter the next round of operation that decreases with the passage of time.

* For a deadline queue whose CT is not reduced to 0, it can receive packets. In detailed, when a node receives a packet to be forwarded from a specific outgoing port, it first obtains the expected deadline of the packet, and then put the packet to the deadline queue with the relevant CT value of the outgoing port for transmission.

* For a deadline queue whose CT is not reduced to 0, its scheduling priority cannot be set to the highest value. The smaller the CT, the higher the priority. A transmission mode can be further configured for a deadline queue to control its transmission behavior. There are two modes:

  - The first mode is to allow participation in priority based scheduling, also termed as in-time mode;

  - The second mode, not allowed, also termed as on-time mode. That is, a queue with on-time mode is allowed to participate in priority based scheduling only when its CT becomes 0.

In-time mode is work-conserving and applicable to low latency services, and on-time mode is not work-conserving and applicable to low latency variation services. Only one mode can be configured for each deadline queue. One implementation can support one set of queues in a single mode, or two sets of queues support two modes.

* At the beginning, all deadline queues have different CT values, i.e., staggered from each other, so that the CT of only one deadline queue will decrease to 0 at any time. It should be noted that CT is just the countdown of the head of the queue, and other packets in the queue (especially those at the end of the queue) have to wait longer to be scheduled. It can be seen that the scheduling countdown of any specific packet in the queue is in the range [CT, CT+AT).

The above AT, TI and MAX_CT value shall be choosed according to the actual capacity of the node. In fact, each node in the network can independently use different AT for different outgoing ports. The general principle is that if an outgoing port has a large bandwidth (such as 100G bps), the AT can be small (such as 1us), because the link with large bandwidth can send the required bits amount even within a small duration; If an outgoing port has a small bandwidth (e.g. 1G bps), the AT should be larger (e.g. 10us), because the link with small bandwidth needs to send the required bits amount within a larger duration. In the FE (Fast Ethernet, 100M bps) scenario, that however may not be the typical scenario this document focuses on, the transmission time of a single packet may take several microseconds, then attention must be paid to ensure that the AT is larger than the transmission time of a single packet, especially the AT should include the interferrence delay caused by a single low priority packet with maximum size.

The choose of TI should consider the latency granularity of various service flows, so that CT updated per TI can match the delay requirements of different services. For example, if the delay difference of different traffic flows is several microseconds, TI can be choosed as 1 us. If the delay difference of different traffic flows is several 10 microseconds, TI can be choosed as 10 us.

A specific example of the deadline queue is depicted in Figure 1.

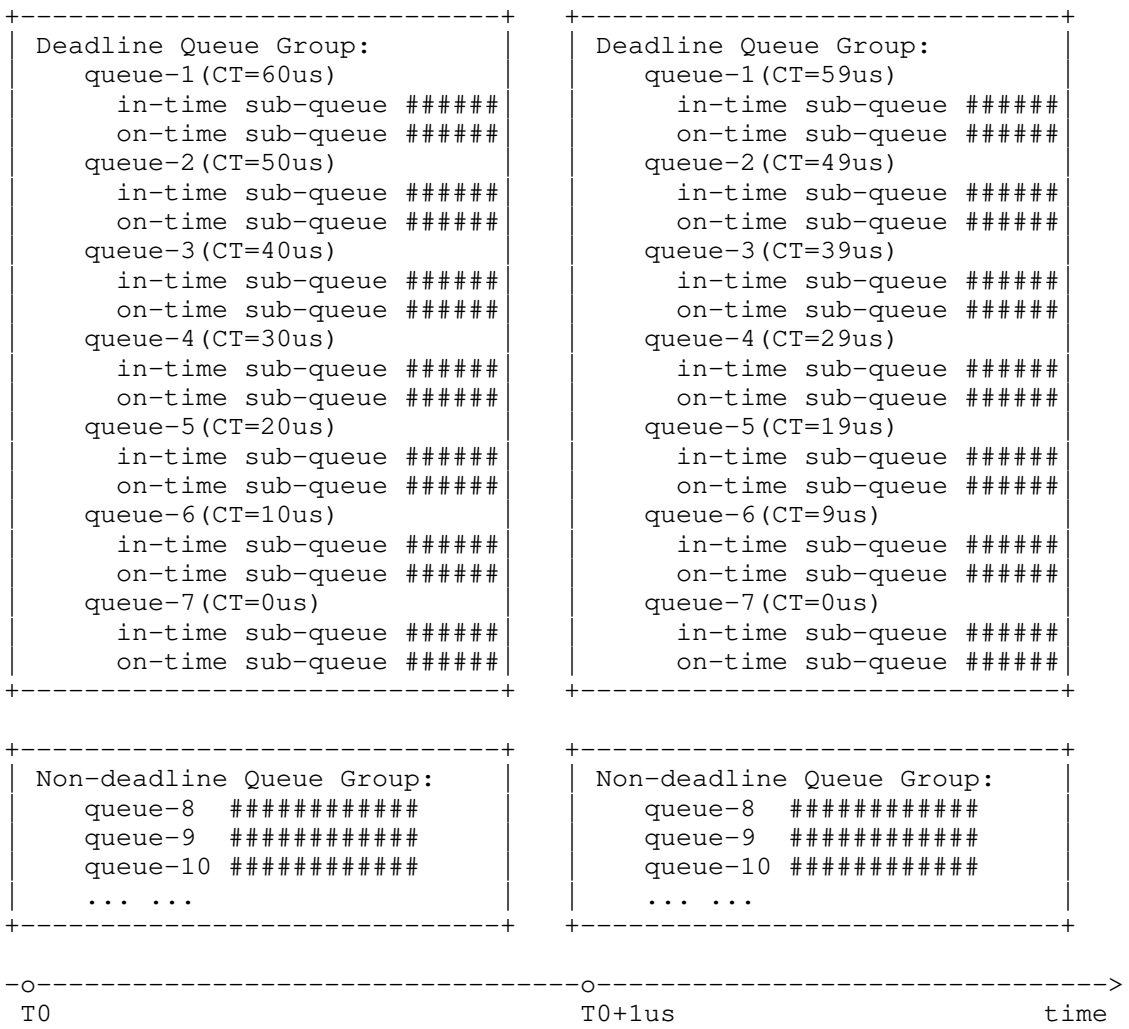


Figure 1: Example of Deadline Queue for outgoing Port

In this example, the AT for deadline queue group is configured to 10us. Queue-1 ~ queue-7 are deadline queues, and other queues are traditional non-deadline queues. Each deadline queue has its CT attribute. The MAX_CT is 60us. At the initial time (T0), the CT of all deadline queues are staggered from each other. For example, the CT of queue-1 is 60us, the CT of queue-2 is 50uS, the CT of queue-3 is 40us, and so on. At this time, only the CT of queue-7 is 0, which has the highest scheduling priority.

Suppose the scheduling engine initiates a cycle timer with a time interval of 1us, i.e., AT = 10 * TI in this case. After each timer timeout, the timer interval will be subtracted from the CT of all deadline queues. As shown in the figure, at T0 + 1us, the timer timeout, the CT of queue-1 becomes 59us, the CT of queue-2 becomes 49us, the CT of queue-3 becomes 39us, etc. At this time, the CT of queue-7, depending on the implementation, can be maintained as 0 or equal to -1.

At T0 + 10us, the CT of queue-1 becomes 50us, the CT of queue-2 becomes 40us, the CT of queue-3 becomes 30us, etc. At this time, the CT of queue-7, returns to MAX_CT and is no longer set to the highest scheduling priority; The CT of queue-6 becomes 0, which has the highest scheduling priority.

When the CT of a deadline queue becomes 0, it has a time limit of 10us (i.e., authorization time) to send packets in the queue. During this period, it will be prohibited to receive new packets (in fact, there can be no new packets with a deadline of 0). After the 10us time elapses, the CT of another deadline queue will change to 0.

If the deadline queue with the highest priority is still free after sending packets within the authorized time, the scheduling engine will visit other queues with the second highest priority during the rest of the authorized time.

Note that if both in-time and on-time policies are supported, the queue-1...7 can be taken regard as logical queues and each logical queue may include two sub-queues, one used to buffer in-time packets and the other used to buffer on-time packets.

3. Get Deadline Information of Packets

3.1. Get Planned Residence Time

The planned residence time of the packet is an offset time, which is based on the time when the packet arrivals the node and represents the maximum time allowed for the packet to stay inside the node. There are many ways to obtain the planned residence time of the packet.

* Carried in the packet. The ingress PE node, when encapsulating the deterministic service flow, can explicitly insert the planned residence time into the packet according to SLA. The intermediate node, after receiving the packet, can directly obtain the planned residence time from the packet. Generally, only a single planned residence time needs to be carried in the packet, which is applicable to all nodes along the path; Or insert a stack composed

of multiple deadlines, one for each node. [I-D.peng-6man-deadline-option] defined a method to carry the planned residence time in the IPv6 packets .

* Included in the FIB entry. Each node in the network can maintain the deterministic FIB entry. After the packet hits the deterministic FIB entry, the planned residence time is obtained from the forwarding information contained in the FIB entry.

* Included in the policy entry. Configure local policies on each node in the network, and then set the corresponding planned residence time according to the matched specific characteristics of the packet, such as 5-tuple. An implementation should support the policy to forcibly override the planned residence time obtained by other methods.

For a deterministic delay path based on deadline scheduling, the path it passes through has deterministic end-to-end delay requirements. It includes two parts, one is the accumulated node delay and the other is the accumulated link propagation delay. The end-to-end delay is subtracted from the accumulated link propagation delay to obtain the accumulated node delay. A simple method is that the accumulated node delay is shared equally by each intermediate node along the path to obtain the planning deadline of each node.

3.2. Get Existing Accumulated Planned Residence Time

The existing accumulated planned residence time of the packet refers to the sum of the planned residence time of all upstream nodes before the packet is transmitted to the current node. This information needs to be carried in the packet. Every time the packet passes through a node, the node accumulates its corresponding planned residence time to the existing accumulated planned residence time field in the packet. [I-D.peng-6man-deadline-option] defined a method to carry existing accumulated planned residence time in the IPv6 packets.

The setting of "existing accumulated planned residence time" in the packet needs to be friendly to the chip for reading and writing. For example, it should be designed as a fixed position in the packet. The chip may support flexible configuration for that position.

3.3. Get Existing Accumulated Actual Residence Time

The existing accumulated actual residence time of the packet, refers to the sum of the actual residence time of all upstream nodes before the packet is transmitted to the current node. This information needs to be carried in the packet. Every time the packet passes through a node, the node accumulates its corresponding actual residence time to the existing accumulated actual residence time field in the packet. [I-D.peng-6man-deadline-option] defined a method to carry existing accumulated actual residence time in the IPv6 packets.

The setting of "existing accumulated actual residence time" in the packet needs to be friendly to the chip for reading and writing. For example, it should be designed as a fixed position in the packet. The chip may support flexible configuration for that position.

Although other methods can also be, for example, carrying the absolute system time of receiving and sending in the packet to compute the actual residence time indirectly, that has a low encapsulation efficiency and require strict time synchronization between nodes.

A possible method to get the actual residence time in the node is that, the receiving and sending time of the packet can be recorded in the auxiliary data structure (note that is not packet itself) of the packet, and the actual residence time of the packet in the node can be calculated according to these two times.

3.4. Get Existing Accumulated Residence Time Deviation

The existing accumulated residence time deviation equals existing accumulated planned residence time minus existing accumulated actual residence time. This value can be positive or negative.

If the existing accumulated planned residence time and the existing accumulated actual residence time are carried in the packet, it is not necessary to carry the existing accumulated residence time deviation. Otherwise, it is necessary. The advantage of the former is that it can be applied to more scenarios, while the later has less packaging overhead.

4. Put Packets into the Deadline Queues

[I-D.ietf-detnet-bounded-latency] presents a latency model for DetNet nodes. There are six type of delays that a packet can experience from hop to hop. The processing delay (type-4), the regulator delay (type-5), the queuing subsystem delay (type-6), and the output delay (type-1) together contribute to the residence time in the node.

In this document, the residence delay in the node is simplified into two parts: the first part is to lookup the forwarding table when the packet is received from the incoming port (or generated by the control plane) and deliver the packet to the line card where the outgoing port is located; the second part is to store the packet in the queue of the outgoing port for transmission. These two parts contribute to the actual residence time of the packet in the node. The former can be called forwarding delay and the latter can be called queuing delay. The forwarding delay is related to the chip implementation and is generally constant; The queuing delay is unstable.

When a node receives a packet from an upstream node, it can first get the existing accumulated residence time deviation, and then add it to the planned residence time of the packet at this node to obtain the deadline adjustment value, and then on the basis of the deadline adjustment value, deducting the forwarding delay of the packet in the node, the allowable queuing delay value (termd as Q) is obtained, and then the packet will be put to the deadline queue with corresponding CT, meeting the condition: CT <= Q < CT+AT. As mentioned above, if the queue already contains many packets, the newly inserted packet may need to wait more time to be scheduled. Therefore, an alternate implemantation may support to select deadline queue for the packet based on Q-k*AT, where 0 <= k <= 1, i.e., meeting the condition: CT <= Q-k*AT < CT+AT. In fact, according to the following introduction, the deadline scheduling mechanism supports latency credit compensation, so the end-to-end latency caused by these two conditions is the same.

Assume that the local node in a deterministic path is i, all upstream nodes are from 1 to i-1, and downstream nodes are i + 1, the planned residence time is D, the actual residence time is R, the deadline adjustment value is M, the forwarding delay inside the node is F, the existing accumulated residence time deviation is E, and the allowable queuing delay is Q, then the allowable queuing delay (Q) of the packet on this node i is calculated as follows:

$$E(i-1) = D(1) + D(2) + ... + D(i-1) - R(1) - R(2) - ... - R(i-1)$$

$$M(i) = D(i) + E(i-1)$$

$$Q(i) = M(i) - F(i)$$

Under normal circumstances, if each hop strictly controls the scheduling of the packet according to its planned residence time, the actual residence time of the packet will be very close to the planned residence time, thus the absolute value of the existing accumulated residence time deviation will be very small.

Consider some extreme cases. For example, many upstream nodes adopt the in-time mode to send packets quickly. Packets almostly need not queue in these nodes, but only depend on the forwarding delay. Then the existing accumulated residence time deviation (E) may be a very large positive value, resulting in a large allowable queuing delay (Q). If this value exceeds the MAX_CT of the deadline queue maintained by the node, the allowable queuing delay (Q) should be modified to the MAX_CT, or such a packet can be inserted into an escape deadline queue with fixed larger CT than MAX_CT, or it can also be inserted into a higher-level queue in a hierarchical deadline queue.

For another example, if some upstream nodes are abnormal and have a very large actual residence time (R), the existing accumulated residence time deviation (E) may be a negative number, resulting in the allowable queuing delay (Q) may be less than or equal to 0, the allowable queuing delay (Q) should be modified to a minimum CT other than 0.

If the queue choosed by the condition is full, the packet must be inserted into the next queue with higher CT value.

Figure 2 depicts an example of packets inserted to the deadline queues.

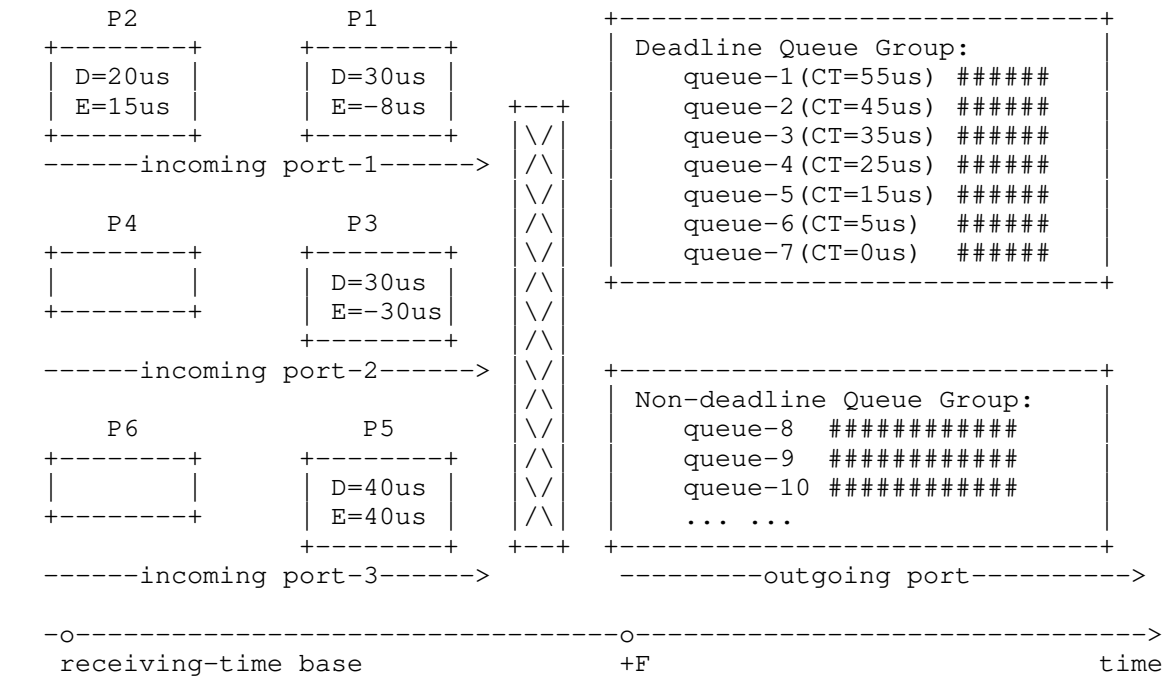


Figure 2: Time Sensitive Packets Buffered to Deadline Queue

As shown in Figure 2, the node successively receives six packets from three incoming ports, among which packet 1, 2, 3 and 5 have corresponding deadline information, while packet 4 and 6 are traditional packets. These packets need to be forwarded to the same outgoing port according to the forwarding table entries. It is assumed that they arrive at the line card where the outgoing port is located at almost the same time after the forwarding delay in the node (F = 5us). At this time, the queue status of the outgoing port is shown in the figure. Then:

* The allowable queuing delay (Q) of packet 1 in the node is 30 - 8 - 5 = 17us, and it will be put into the deadline queue-5 (its CT is 15us), meeting the condition that Q is in the range [15, 25).

* The allowable queuing delay (Q) of packet 2 in the node is 20 + 15 - 5 = 30us, and it will be put into the deadline queue-4 (its CT is 25us), meeting the condition that Q is in the range [25, 35).

* The allowable queuing delay (Q) of packet 3 in the node is 30 - 30 - 5 = -5us, and it will be modified to the minimum positive value 5 us then put into the deadline queue-6 (its CT is 5us), meeting the condition that Q is in the range [5, 15).

* The allowable queuing delay (Q) of packet 5 in the node is 40 + 40 - 5 = 75us, and it will be modified to the MAX_CT (60 us) then put into the deadline queue-1 (its CT is 55us), meeting the condition that Q is in the range [55, 65).

* Packets 4 and 6 will be put into the non-deadline queue in the traditional way.

4.1. Alternate Queue Allocation Ruels

Each deadline queue can be further divided into multiple sub-queues, and each planned residence time corresponds to one. That is, packets with different planned residence time are inserted into different sub-queues and protected. In this way, for two packets with the same allowable queuing delay (Q) but different planned residence time, we can decide to schedule the packet with the smaller planned residence time. This is beneficial because when packets with smaller planned residence time facing larger interference delay, it is difficult to have space for delay compensation on downstream nodes, while packets with larger planned residence time have larger space for delay compensation.

5. Buffer Size Design

We assume that the service rate of the deadline scheduler can be the link rate (termed as C), so the buffer size of each deadline queue is AT * C - M, where M is the maximum size of the packet with low priority.

6. Traffic Regulation and Orchestrating

On the ingress PE node, traffic regulation should be performed on the incoming port, so that the service traffic does not exceed its constraint funciton, such as leacky bucket $A_i(t) = b_i + r_i*t$ for service i. Suppose there are multiple service flows released to the network through the ingress node, then it must meet $sum\{r_i\} <= C$, for all service i and any time t, where C is service rate of the deadline scheduler. In addition, stability condition $sum\{A_i(t)\}/t < C$ must also be met.

Then, on the ingress PE node, traffic is orchestrated into different deadline queues of the outgoing port. Multiple continuous packets of the specific service flow are stored in the deadline queue with corresponding remaining time according to the planned residence time of the service flow. Note that these packets are not stored in the same queue over time. The amount of bits that can be stored in one queue for service i is equal to $R_i * AT$, however, at least one whole packet shall be loaded. For example, if the allowable queuing delay

(Q) is 20us, then within the current authorization time, the first sequence of the packets will be put into the current deadline queue with [CT, CT+AT) range that can cover 20us, until the reserved bandwidth limit is reached; Then, within the next authorization time, the next sequence of packets will be put into the queue with [CT, CT+AT) range that can cover 20us, until the reserved bandwidth limit is reached; and so on, until the total service bits are loaded.

Figure 3 depicts an example of deadline based traffic orchestrated on the ingress PE node. It is assumed that the packets loaded in each authorization time do not exceed the reserved bandwidth of the service.

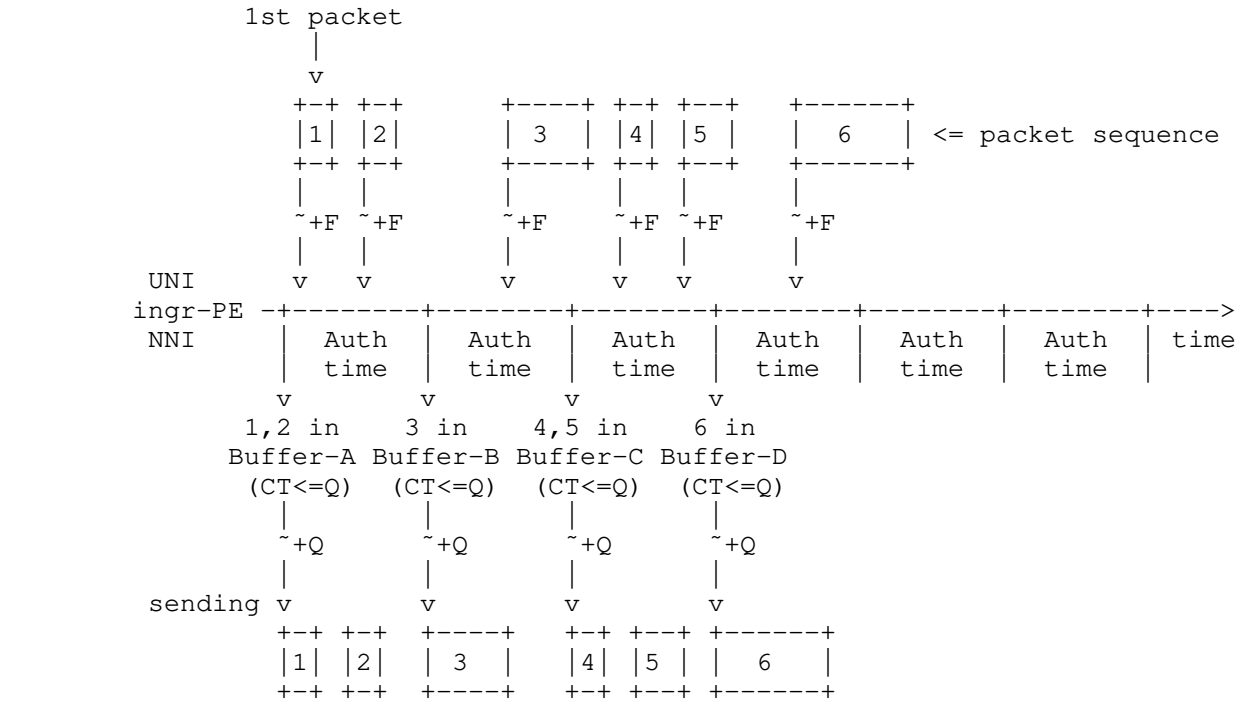


Figure 3: Deadline Based Packets Orchestrating

6.1. Traffic Re-shaping on Intermediate Node

Multiple ingress PEs may release multiple flows with the same deadline. These flows may arrive at an intermediate node at the same time and put into the same deadline queue. Especially, packets with different deadlines sent by a single ingress PE node at different instant of time (e.g, for on-time mode the packet sent early has a larger planned residence time than the packet sent late, or for in-time mode the packet sent early face more interference delay than the packet sent late.), may be put into the same deadline queue by an intermediate node. This means that a larger bandwidth is required on the intermediate node than the ingress PE node to send more bits within the same time duration, which also means more buffer size.

Deadline in-time mode is a variant of EDF scheduler with work-conserving characteristic and has several aggregated queues. It needs to rely on the known traffic arrival curve to obtain sufficient and necessary schedulability conditions. However, in traffic aggregation case, the arrival traffic faced by nodes may be irregular due to work-conserving scheduling. One solution is to re-shape the traffic on each intermediate node, such as per flow re-shaping or interleaved regulator introduced by [UBS]. It is not recommended to maintain per flow shapers in large-scale network. After placing re-shapers in front of the deadline scheduler, the combination of the re-shaper and the deadline scheduler is no longer work-conserving. As described in [IR-Theory], per flow shaper or interleaved regulator does not increate worst-case latency bounds. Whether this conclusion is still valid for deadline needs further study. Suppose that after re-shaping, for an in-time service i, the arrival curve is A_i(t), and the planned residence time is d_i which meeting d_i < d_(i+1) of service i+1, then the schedulability condition for deadline in-time mode is:.

$$A_1(t-d_1) + \sum\{A_i(t+AT-d_i) \text{ for all } i>=2\} <= C*t$$

where AT is authorization time of each deadline queue, C is service rate of the deadline scheduler.

The proof is similar with that in [RPQ], except that the step length is fine-grained when the queue rotates. Figure 4 below gives a rough explanation.

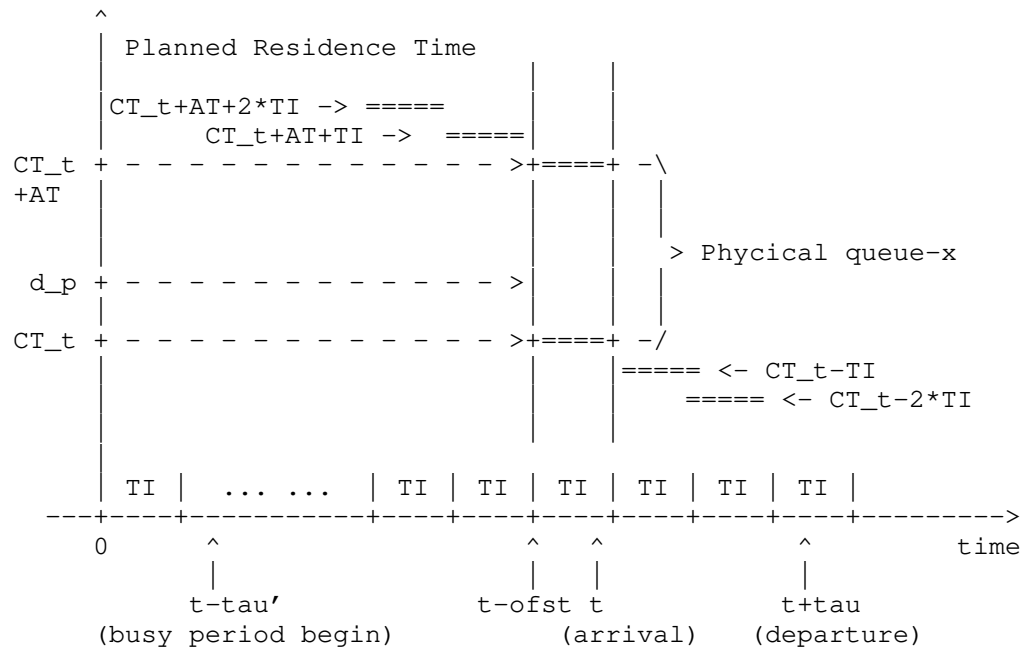


Figure 4: Deadline In-time Scheduling

Suppose that the observed packet, with planned residence time d_p, arrives at the scheduler at time t and leaves the scheduler at time t+tau. It will be inserted to physical queue-x with count-down time CT_t at the current timer interval TI with starting time t-ofst and end time t-ofst+TI. According to the above packet queuing rules, we have CT_t <= d_p < CT_t+AT. Also suppose that t-tau' is the beginning of the busy period closest to t. Then, we can get the amount of packets within time interval [t-tau', t+tau] that must be scheduled before the observed packet. In detailed:

for all service i with planned residence time d_i meeting CT_t <= d_i < CT_t+AT, the workload is sum{A_i[t-tau', t]}.

for all service i with planned residence time d_i meeting d_i >= CT_t+AT, the workload is sum{A_i[t-tau', t-ofst+TI-(d_i-CT_t-AT+TI)]}.

for all service i with planned residence time d_i meeting d_i < CT_t, the workload is sum{A_i[t-tau', t+(CT_t-d_i)]}.

then deduct the traffic that has been sent during the busy period, i.e., C*(tau+tau').

Let tau as d_p, and remember that CT_t <= d_p, the above workload is less than

sum{A_i(tau'+CT_t+AT-d_i) for all d_i >= CT_t} +
sum{A_i(tau'+CT_t-d_i) for all d_i < CT_t} - C*(tau'+d_p)

It is further less than

sum{A_i(tau'+d_p+AT-d_i) for all d_i >= d_2} + A_1(tau'+d_p-d_1) - C*(tau'+d_p)

Then, denote x as tau'+d_p, we have

sum{A_i(x+AT-d_i) for all d_i >= d_2} + A_1(x-d_1) - C*(x)

Let the above workload less than zero, then we get the condition.

It should be noted that for a service i with planned residence time d_i, its planned residence time is actually contributed by its own flow and all the flows with lower planned residence time. Thus, based on the above schedulability conditions, and knowing the traffic constraint functions of all services, we can then give the guidance to allocate appropriate (i.e., not arbitrary) planned residence time for each service. In brief, we can choose the traffic arrival constraint function according to the preset planned residence time, or we can choose the planned residence time according to the preset traffic arrival constraint function.

Compared with the in-time mode, on-time mode is non-work-conserving, which can be considered as the combination of EDF and damper. The above schedulability conditions can also be applied to the on-time mode, that is, if the aggregate traffic for the specific planned residence time reaches the maximum limit allowed by the conditions, it will achieve passive on-time behavior. If the aggregate traffic is lower than the limit, the deadline on-time scheduler will also achieve active on-time behavior by applying the built-in damper. Since the built-in dumper has shaped the on-time service flow, there is no need to re-shape the on-time service flow by additional shapers in front of deadline scheduler.

7. Delay Compensation Considerations

As mentioned above, the calculation of allowable queuing delay (Q) reflects the meaning of delay compensation. It can be used for in-time services to distinguish between emergency flows and non emergency flows (compared with traditional strict priority scheduling), and also for on-time services to strictly control delay jitter. In both cases, packets have a tolerable maximum queuing

delay in the queuing subsystem. The traditional EDF scheduling can also benefit from this delay compensation.

Suppose that two packets, P1, P2, are generated instantaneously from a specific in-time service flow at the same source, and the two packets have the same planned residence time. Packet-1 may face less interference delay than P2 in their process of forwarding. When they arrive at an intermediate node successively, P2 will have less allowable queuing delay (Q) than P1 to try to stay close to P1 again. It should be noted that to compary who is ealier is based on queue's CT and packet's Q, according to the above queuing rule (CT <= Q < CT+AT), and the CT of the queue is not changed in real-time, but gradually with the decreasing step TI. It is possible to get an unexpected comparision result.

As shown in Figure 5, P1 and P2 are two packets belonging to the same burst. The arrival time when they are received on the current scheduler is shown in the figure. Suppose that the AT of the deadline queue is 10us, the decreasing step TI is 1us, and the transmission time of each packet is 0.01us. Also suppose that the Q values of two adjacent packets P1 and P2 are 40us and 39us respectively, and they are both received in the window from T0 to T0+1us. P1 will enter queue-B with CT range [40, 50), while P2 will enter queue-A with CT range [30, 40). This means that P2 will be scheduled before P1, resulting in disorder.

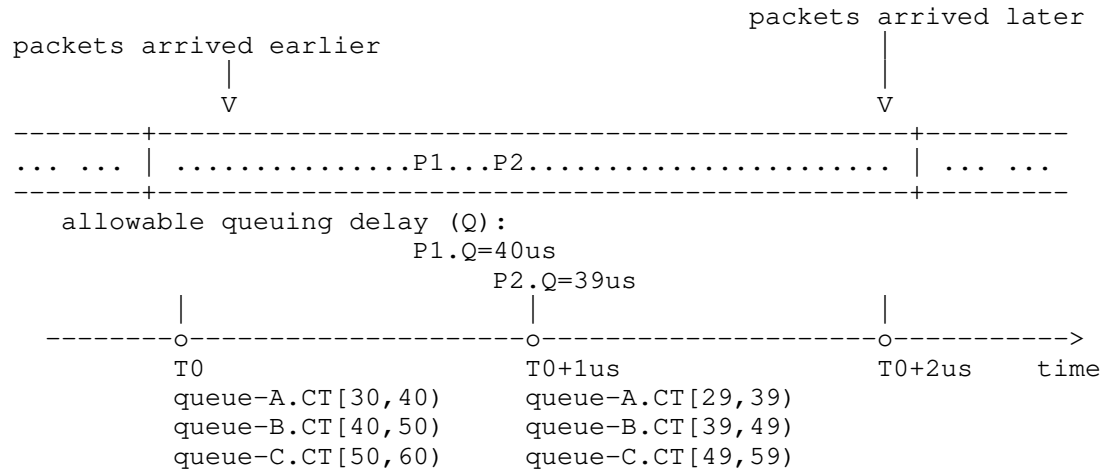


Figure 5: Packets queuing based on Delay Compensation

DetNet architecture [RFC8655] provides Packet Ordering Function, that can be used to solve the above disorder problem caused by the delay compensation. In addition, we can also maintain states for service flows to record the last queuing information to address this issue.

8. Compatibility Considerations

For a particular path, if only some nodes in the path upgrade support the deadline mechanism defined in this document, the end-to-end deterministic delay/jitter target will only be partially achieved. Those legacy devices may adopt the existing priority based scheduling mechanism, and ignore the possible deadline information in the packet, thus the delay intra node produced by them cannot be perceived by the adjacent upgraded node. The more upgraded nodes included in the path, the closer to the delay/jitter target. Although, the legacy devices may not support the dataplane mechanism described in this document, but they can be freely programmed (such as P4 language) to measure and insert the deadline information into packets, in this case the achievement of delay/jitter target will be more perfect.

Only a few key nodes are upgraded to support deadline mechanism, which is low-cost, but can meet a service with relatively loose time requirements. Figure 6 shows an example of upgrading only several network border nodes. In the figure, only R1, R2, R3 and R4 are upgraded to support deadline mechanism. A deterministic path across domain 1, 2, and 3 is established, which contains nodes R1, R2, R3, and R4, as well as explicit nodes in each domain. Domain 1, 2 and 3 use the traditional strict priority based forwarding mechanism. The encoding of the packet sent by R1 includes the planned residence time and the accumulated residence time deviation. Especially, IP DSCP or Traffic Class are also set to appropriate values. The basic principle of setting is that the less the planned residence time, the higher the priority. However, in order to avoid the interference of non deterministic flow to deterministic flow, the priority of deterministic flow should be set as high as possible.

The delay analysis based on strict priority in each domain can be found in [SP-LATENCY], which gives the formula to evaluate the worst-case delay of each hop during the resource reservation procedure. The worst-case delay depends on the number of hops and the burst size of interference flows that may be faced on each hop. [EF-FIFO] also shows that, for FIFO packet scheduling be used to support the EF (expedited forwarding) per-hop behavior (PHB), if the network utilization level alpha < l/(H-l), the worst-case delay bound is inversely proportional to 1-alpha*(H-1), where H is the number of hops in the longest path of the network. Although the deadline in-time scheduling, the SP scheduling and EF FIFO scheduling are all

work-conserving, the deadline in-time scheduling can further distinguish between emergency and non emergency according to deadline information other than traffic class. Therefore, when analyzing the latency of the in-time mode, the latency is not evaluated just according to the order in which the packets arrive at the scheduler, but according to the deadline of the packets. An intuitive phenomenon is that if a packet unfortunately faces more interference delays at the upstream nodes, it will become more urgent at the downstream node, and will not always be unfortunate. This operation of dynamically modifying the key fields of the packet can avoid always overestimating worst-case latency on all hops. According to schedulability condition, in-time mode can be pessimistic that its worst-case latancy bounds is the same as on-time mode.

When the boundary node (e.g, R2) receives the deterministic traffic, it will decide whether to speed up or hold according to the accumulated residence time deviation information carried in the packet. The in-time traffic is always sent as soon as possible, especially when the residence time deviation of the packet is negative. The on-time traffic always controls the sending time so that the average planned residence time is followed, especially when the residence time deviation of the packet is positive. For a specific deterministic flow, if it experiences too much latency in the SP domain (due to unreasonable setting of traffic class and the inability to distinguish between deterministic and non deterministic flows), even if the boundary node accelerates the transmission, it may not be able to achieve the target of low E2E latency. If the traffic experiences less latency within the SP domain, on-time mode will work to achieve the end-to-end jitter target.

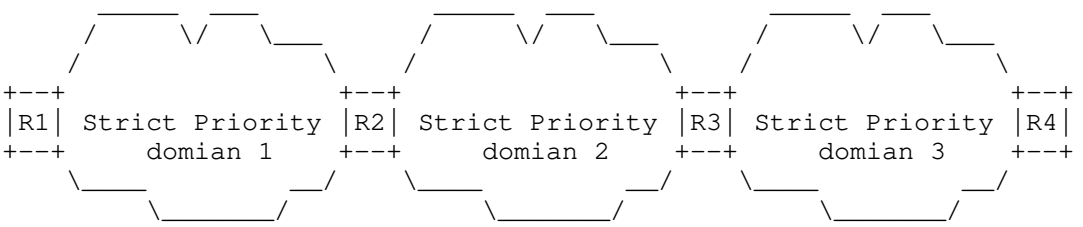


Figure 6: Example of partial upgrade

9. Deployment Considerations

According to the above schedulability conditions, the planned residence time that can be provided in the network is related to the entire deployed service flows. Each planned residence time can be considered as a delay resource, and the smaller the residence time, the more valuable it is. The operator needs to match the corresponding planned residence time for each service. It is not recommended to allocate a large planned residence time for the on-time service, which will consume a large amount of buffer and is not beneficial to target jitter. However, the in-time service may also not want to be assigned to a larger planned residence time, that may be determined by its SLA that requires a smaller target end-to-end latency. Thus it is necessary to find a balance to meet more service requirements as much as possible.

We believe that the planned residence time specified for the service flows is completely determined by the network, not by the user. Therefore, it is impossible to carry the planned residence time beyond the network capacity in the service packets. However, there may be some services that expect a large residence time, in this case it is recommended to hold them at the border nodes, but not on the intermediate node. For this purpose, the MAX_CT configured on the border nodes can be much larger than the MAX_CT configured on the intermediate nodes, or the hierarchical deadline queues can be configured on the border nodes.

10. Benefits

The mechanism described in this document has the following benefits:

* Time synchronization is not required between network nodes. Each node can flexibly set the authorization time length of the deadline queue according to its own outgoing port bandwidth.

* Packet multiplexing based, it is an enhancement of PQ scheduling algorithm, friendly to the upgrade of packet switching network. All nodes in the network can independently use cycle timers with different timeout intervals to rotate the deadline queues.

* The packet can control its expected residence time in the node. A single set of deadline queues supports multiple levels of residence time.

* For in-time mode, the end-to-end delay is H*(F~D), jitter is H*Q; For on-time mode, the end-to-end delay is H*D, jitter is a just single authorization time.

11. IANA Considerations

There is no IANA requestion for this document.

12. Security Considerations

TBD

13. Acknowledgements

TBD

14. References

14.1. Normative References

[I-D.ietf-detnet-bounded-latency]
Finn, N., Boudec, J. L., 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.peng-6man-deadline-option]
Peng, S., Tan, B., and P. Liu, "Deadline Option", Work in Progress, Internet-Draft, draft-peng-6man-deadline-option-01, 11 July 2022, <https://www.ietf.org/archive/id/draft-peng-6man-deadline-option-01.txt>.

[I-D.stein-srtsn]
Stein, Y. (., "Segment Routed Time Sensitive Networking", Work in Progress, Internet-Draft, draft-stein-srtsn-01, 29 August 2021, <https://www.ietf.org/archive/id/draft-stein-srtsn-01.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

14.2. Informative References

[EF-FIFO] "Fundamental Trade-Offs in Aggregate Packet Scheduling", 2001, <https://ieeexplore.ieee.org/document/992892>.

[IR-Theory] "A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators", 2018, <https://ieeexplore.ieee.org/document/8519761>.

[RPQ] "Exact Admission Control for Networks with a Bounded Delay Service", 1996, <https://ieeexplore.ieee.org/document/556345/>.

[SP-LATENCY] "Guaranteed Latency with SP", 2020, <https://www.ieee802.org/1/files/public/docs2020/dd-grigorjew-strict-priority-latency-0320-v02.pdf>.

[UBS] "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks", 2016, <https://ieeexplore.ieee.org/abstract/document/7557870>.

Authors' Addresses

Shaofu Peng
ZTE Corporation
China
Email: peng.shaofu@zte.com.cn

Bin Tan
ZTE Corporation
China
Email: tan.bin@zte.com.cn

Peng Liu
China Mobile
China
Email: liupengyjy@chinamobile.com

Network
Internet-Draft
Intended status: Standards Track
Expires: 11 June 2023

Shaofu. Peng
Bin. Tan
ZTE Corporation
Peng. Liu
China Mobile

8 December 2022

Deadline Based Deterministic Forwarding
draft-peng-detnet-deadline-based-forwarding-04

Abstract

This document describes a deterministic forwarding mechanism based on deadline. The mechanism enhances strict priority scheduling algorithm with dynamically adjusting the priority of the queue according to its deadline attribute.

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 11 June 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

[RFC8655] describes the architecture of deterministic network and defines the QoS goals of deterministic forwarding: Minimum and maximum end-to-end latency from source to destination, timely delivery, and bounded jitter (packet delay variation); packet loss ratio under various assumptions as to the operational states of the nodes and links; an upper bound on out-of-order packet delivery. In order to achieve these goals, deterministic networks use resource reservation, explicit routing, service protection and other means. Resource reservation refers to the occupation of resources by service traffic, exclusive or shared in a certain proportion, such as dedicated physical link, link bandwidth, queue resources, etc; Explicit routing means that the transmission path of traffic flow in the network needs to be selected in advance to ensure the stability of the route and does not change with the real-time change of network topology, and based on this, the upper bound of end-to-end delay and delay jitter can be accurately calculated; Service protection refers to sending multiple service flows along multiple disjoint paths at the same time to reduce the packet loss rate. In general, a

deterministic path is a strictly explicit path calculated by a centralized controller, and resources are reserved on the nodes along the path to meet the SLA requirements of deterministic services.

[I-D.stein-srtsn] describes that the controller calculates the local deadline time of each node for the traffic to be transmitted in advance, which is a absolute system time, forms a stack of these local deadline time, and then carries them in the forwarded data packets. Each node forwards the packets according to its own local deadline. [I-D.stein-srtsn] suggests that FIFO queue can not be used to realize this function, because the packets stored in the queue are always first in first out and a special data structure is recommoned. The packets in this data structure will be automatically sorted with the order from emergency to non emergency according to the deadline of the packets. However, it may be difficult to implement this structure in hardware, and especially for a large network it may be challenge to synchronize time.

Considering that the link propagation delay is generally a fixed value, and we focus on the residence time of the packets inside the node, an alternate approach is to make the deadline eliminate the interference of link propagation delay and avoids relying on time synchronization between nodes.

This document desrbies an alternate packets scheduling scheme based on EDF (earliest-deadline-firs) combined with latency compensation and used for wide area network. It suggests to only use a single deadline time to control the packets scheduling of all nodes along the path. The single deadline time is an offset time, which is based on the time when the packet enters the node and represents the maximum time allowed for the packet to stay inside the node. However, if each node has obvious differences in the capability of packets forwarding and scheduling, more offset-time may be needed.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Deadline Queue

For nodes in the network, some queues with count-down time (also termed as CT) are introduced and maintained for each outgoing port. These queues are called deadline queues and participate in priority based scheduling. Deadline queues have the following characteristics:

* The CT of each deadline queue will decrease with the passage of time. When it decreases to 0, the scheduling priority of the queue will be set to the highest, and the scheduling opportunity will be obtained immediately (note that there may be interference delay caused by a packet being sent by a low priority queue). It will prohibit receiving new packets, in which the buffered packets will be sent to the outgoing port immediately, and the maximum duration allowed to send packets is the preset authorization time (also termed as AT), e.g, 10us, 20us, etc. In principle, all packets buffered in the queue shall be sent within this authorization time. If the queue is sent to empty and the authorization time is still free, other queues with lower priority can be scheduled during this authorization time.

* The scheduling engine can initiate a cycle timer to decrement the CT of all deadline queues, that is, whenever the timer expires, the CT values of all queues will be subtracted from the timer interval (also termed as TI). Note that TI must be less than or equal to the AT, with AT = N * TI, where the natural number N >= 1.

* For a deadline queue whose CT has been reduced to 0, after a new round of authorization time, the CT will return to the maximum initial value (also termed as MAX_CT), allow receiving new packets, and continue to enter the next round of operation that decreases with the passage of time.

* For a deadline queue whose CT is not reduced to 0, it can receive packets. In detailed, when a node receives a packet to be forwarded from a specific outgoing port, it first obtains the expected deadline of the packet, and then put the packet to the deadline queue with the relevant CT value of the outgoing port for transmission.

* For a deadline queue whose CT is not reduced to 0, its scheduling priority cannot be set to the highest value. The smaller the CT, the higher the priority. A transmission mode can be further configured for a deadline queue to control its transmission behavior. There are two modes:

- The first mode is to allow participation in priority based scheduling, also termed as in-time mode;

- The second mode, not allowed, also termed as on-time mode. That is, a queue with on-time mode is allowed to participate in priority based scheduling only when its CT becomes 0.

In-time mode is work-conserving and applicable to low latency services, and on-time mode is not work-conserving and applicable to low latency variation services. Only one mode can be configured for each deadline queue. One implementation can support one set of queues in a single mode, or two sets of queues support two modes.

* At the beginning, all deadline queues have different CT values, i.e., staggered from each other, so that the CT of only one deadline queue will decrease to 0 at any time. It should be noted that CT is just the countdown of the head of the queue, and other packets in the queue (especially those at the end of the queue) have to wait longer to be scheduled. It can be seen that the scheduling countdown of any specific packet in the queue is in the range [CT, CT+AT).

The above AT, TI and MAX_CT value shall be choosed according to the actual capacity of the node. In fact, each node in the network can independently use different AT for different outgoing ports. The general principle is that if an outgoing port has a large bandwidth (such as 100G bps), the AT can be small (such as 1us), because the link with large bandwidth can send the required bits amount even within a small duration; If an outgoing port has a small bandwidth (e.g. 1G bps), the AT should be larger (e.g. 10us), because the link with small bandwidth needs to send the required bits amount within a larger duration. In the FE (Fast Ethernet, 100M bps) scenario, that however may not be the typical scenario this document focuses on, the transmission time of a single packet may take several microseconds, then attention must be paid to ensure that the AT is larger than the transmission time of a single packet, especially the AT should include the interferrence delay caused by a single low priority packet with maximum size.

The choose of TI should consider the latency granularity of various service flows, so that CT updated per TI can match the delay requirements of different services. For example, if the delay difference of different traffic flows is several microseconds, TI can be choosed as 1 us. If the delay difference of different traffic flows is several 10 microseconds, TI can be choosed as 10 us.

A specific example of the deadline queue is depicted in Figure 1.

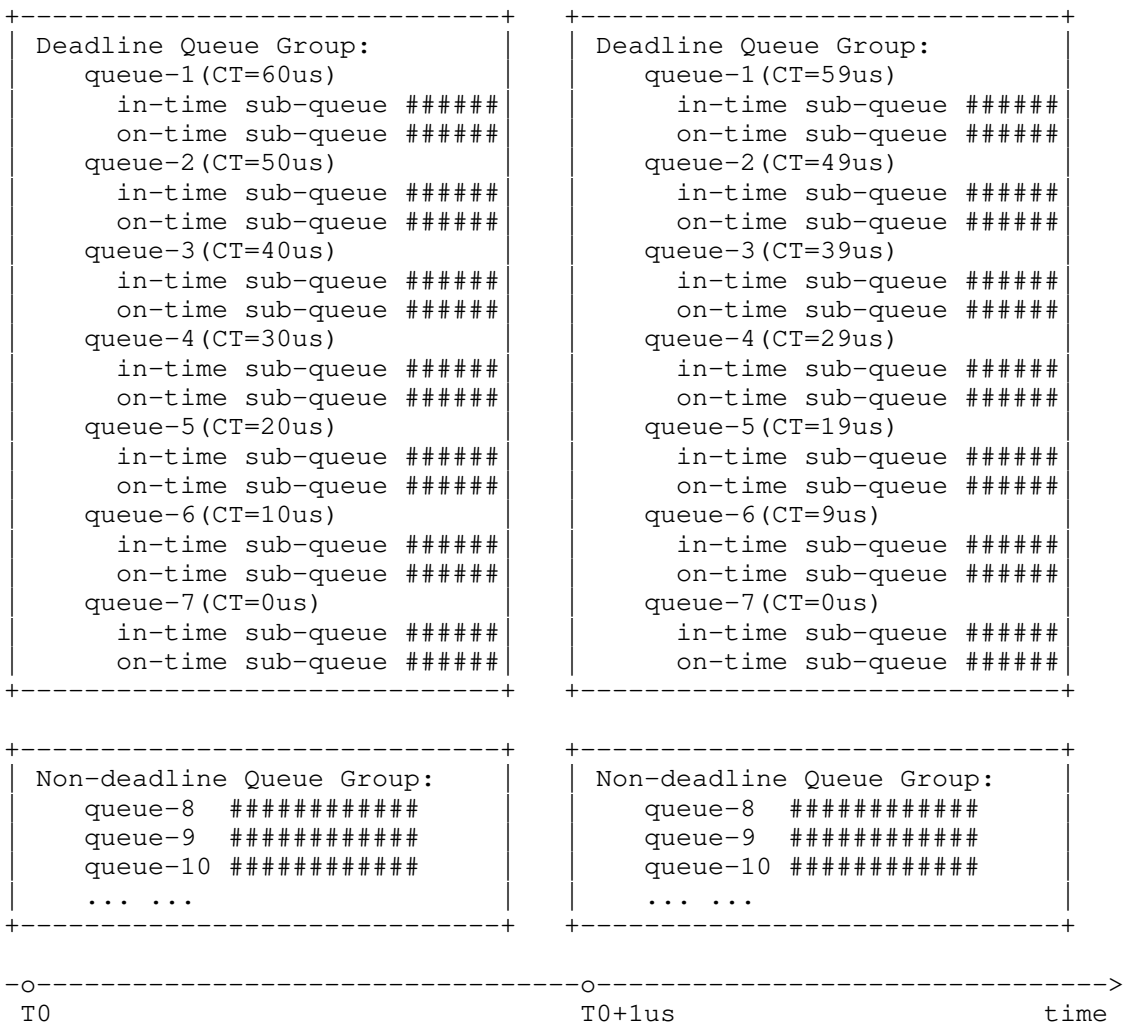


Figure 1: Example of Deadline Queue for outgoing Port

In this example, the AT for deadline queue group is configured to 10us. Queue-1 ~ queue-7 are deadline queues, and other queues are traditional non-deadline queues. Each deadline queue has its CT attribute. The MAX_CT is 60us. At the initial time (T0), the CT of all deadline queues are staggered from each other. For example, the CT of queue-1 is 60us, the CT of queue-2 is 50uS, the CT of queue-3 is 40us, and so on. At this time, only the CT of queue-7 is 0, which has the highest scheduling priority.

Suppose the scheduling engine initiates a cycle timer with a time interval of 1us, i.e., AT = 10 * TI in this case. After each timer timeout, the timer interval will be subtracted from the CT of all deadline queues. As shown in the figure, at T0 + 1us, the timer timeout, the CT of queue-1 becomes 59us, the CT of queue-2 becomes 49us, the CT of queue-3 becomes 39us, etc. At this time, the CT of queue-7, depending on the implementation, can be maintained as 0 or equal to -1.

At T0 + 10us, the CT of queue-1 becomes 50us, the CT of queue-2 becomes 40us, the CT of queue-3 becomes 30us, etc. At this time, the CT of queue-7, returns to MAX_CT and is no longer set to the highest scheduling priority; The CT of queue-6 becomes 0, which has the highest scheduling priority.

When the CT of a deadline queue becomes 0, it has a time limit of 10us (i.e., authorization time) to send packets in the queue. During this period, it will be prohibited to receive new packets (in fact, there can be no new packets with a deadline of 0). After the 10us time elapses, the CT of another deadline queue will change to 0.

If the deadline queue with the highest priority is still free after sending packets within the authorized time, the scheduling engine will visit other queues with the second highest priority during the rest of the authorized time.

Note that if both in-time and on-time policies are supported, the queue-1...7 can be taken regard as logical queues and each logical queue may include two sub-queues, one used to buffer in-time packets and the other used to buffer on-time packets.

3. Get Deadline Information of Packets

3.1. Get Planned Residence Time

The planned residence time of the packet is an offset time, which is based on the time when the packet arrivals the node and represents the maximum time allowed for the packet to stay inside the node. There are many ways to obtain the planned residence time of the packet.

* Carried in the packet. The ingress PE node, when encapsulating the deterministic service flow, can explicitly insert the planned residence time into the packet according to SLA. The intermediate node, after receiving the packet, can directly obtain the planned residence time from the packet. Generally, only a single planned residence time needs to be carried in the packet, which is applicable to all nodes along the path; Or insert a stack composed

of multiple deadlines, one for each node. [I-D.peng-6man-deadline-option] defined a method to carry the planned residence time in the IPv6 packets .

* Included in the FIB entry. Each node in the network can maintain the deterministic FIB entry. After the packet hits the deterministic FIB entry, the planned residence time is obtained from the forwarding information contained in the FIB entry.

* Included in the policy entry. Configure local policies on each node in the network, and then set the corresponding planned residence time according to the matched specific characteristics of the packet, such as 5-tuple. An implementation should support the policy to forcibly override the planned residence time obtained by other methods.

For a deterministic delay path based on deadline scheduling, the path it passes through has deterministic end-to-end delay requirements. It includes two parts, one is the accumulated node delay and the other is the accumulated link propagation delay. The end-to-end delay is subtracted from the accumulated link propagation delay to obtain the accumulated node delay. A simple method is that the accumulated node delay is shared equally by each intermediate node along the path to obtain the planning deadline of each node.

3.2. Get Existing Accumulated Planned Residence Time

The existing accumulated planned residence time of the packet refers to the sum of the planned residence time of all upstream nodes before the packet is transmitted to the current node. This information needs to be carried in the packet. Every time the packet passes through a node, the node accumulates its corresponding planned residence time to the existing accumulated planned residence time field in the packet. [I-D.peng-6man-deadline-option] defined a method to carry existing accumulated planned residence time in the IPv6 packets.

The setting of "existing accumulated planned residence time" in the packet needs to be friendly to the chip for reading and writing. For example, it should be designed as a fixed position in the packet. The chip may support flexible configuration for that position.

3.3. Get Existing Accumulated Actual Residence Time

The existing accumulated actual residence time of the packet, refers to the sum of the actual residence time of all upstream nodes before the packet is transmitted to the current node. This information needs to be carried in the packet. Every time the packet passes through a node, the node accumulates its corresponding actual residence time to the existing accumulated actual residence time field in the packet. [I-D.peng-6man-deadline-option] defined a method to carry existing accumulated actual residence time in the IPv6 packets.

The setting of "existing accumulated actual residence time" in the packet needs to be friendly to the chip for reading and writing. For example, it should be designed as a fixed position in the packet. The chip may support flexible configuration for that position.

Although other methods can also be, for example, carrying the absolute system time of receiving and sending in the packet to compute the actual residence time indirectly, that has a low encapsulation efficiency and require strict time synchronization between nodes.

A possible method to get the actual residence time in the node is that, the receiving and sending time of the packet can be recorded in the auxiliary data structure (note that is not packet itself) of the packet, and the actual residence time of the packet in the node can be calculated according to these two times.

3.4. Get Existing Accumulated Residence Time Deviation

The existing accumulated residence time deviation equals existing accumulated planned residence time minus existing accumulated actual residence time. This value can be positive or negative.

If the existing accumulated planned residence time and the existing accumulated actual residence time are carried in the packet, it is not necessary to carry the existing accumulated residence time deviation. Otherwise, it is necessary. The advantage of the former is that it can be applied to more scenarios, while the later has less packaging overhead.

4. Put Packets into the Deadline Queues

[RFC9320] presents a latency model for DetNet nodes. There are six type of delays that a packet can experience from hop to hop. The processing delay (type-4), the regulator delay (type-5), the queueing subsystem delay (type-6), and the output delay (type-1) together contribute to the residence time in the node.

In this document, the residence delay in the node is simplified into two parts: the first part is to lookup the forwarding table when the packet is received from the incoming port (or generated by the control plane) and deliver the packet to the line card where the outgoing port is located; the second part is to store the packet in the queue of the outgoing port for transmission. These two parts contribute to the actual residence time of the packet in the node. The former can be called forwarding delay and the latter can be called queueing delay. The forwarding delay is related to the chip implementation and is generally constant; The queueing delay is unstable.

When a node receives a packet from an upstream node, it can first get the existing accumulated residence time deviation, and then add it to the planned residence time of the packet at this node to obtain the deadline adjustment value, and then on the basis of the deadline adjustment value, deducting the forwarding delay of the packet in the node, the allowable queueing delay value (termd as Q) is obtained, and then the packet will be put to the deadline queue with corresponding CT, meeting the condition: CT <= Q < CT+AT. As mentioned above, if the queue already contains many packets, the newly inserted packet may need to wait more time to be scheduled. Therefore, an alternate implemantation may support to select deadline queue for the packet based on Q-k*AT, where 0 <= k <= 1, i.e., meeting the condition: CT <= Q-k*AT < CT+AT. In fact, according to the following introduction, the deadline scheduling mechanism supports latency credit compensation, so the end-to-end latency caused by these two conditions is the same.

Assume that the local node in a deterministic path is i, all upstream nodes are from 1 to i-1, and downstream nodes are i + 1, the planned residence time is D, the actual residence time is R, the deadline adjustment value is M, the forwarding delay inside the node is F, the existing accumulated residence time deviation is E, and the allowable queueing delay is Q, then the allowable queueing delay (Q) of the packet on this node i is calculated as follows:

* E(i-1) = D(1) + D(2) + ... + D(i-1) - R(1) - R(2) - ... - R(i-1)

* M(i) = D(i) + E(i-1)

* Q(i) = M(i) - F(i)

Under normal circumstances, if each hop strictly controls the scheduling of the packet according to its planned residence time, the actual residence time of the packet will be very close to the planned residence time, thus the absolute value of the existing accumulated residence time deviation will be very small.

Consider some extreme cases. For example, many upstream nodes adopt the in-time mode to send packets quickly. Packets almostly need not queue in these nodes, but only depend on the forwarding delay. Then the existing accumulated residence time deviation (E) may be a very large positive value, resulting in a large allowable queueing delay (Q). If this value exceeds the MAX_CT of the deadline queue maintained by the node, the allowable queueing delay (Q) should be modified to the MAX_CT, or such a packet can be inserted into an escape deadline queue with fixed larger CT than MAX_CT, or it can also be inserted into a higher-level queue in a hierarchical deadline queue.

For another example, if some upstream nodes are abnormal and have a very large actual residence time (R), the existing accumulated residence time deviation (E) may be a negative number, resulting in the allowable queueing delay (Q) may be less than or equal to 0, the allowable queueing delay (Q) should be modified to a minimum CT other than 0.

If the queue choosed by the condition is full, the packet must be inserted into the next queue with higher CT value.

Figure 2 depicts an example of packets inserted to the deadline queues.

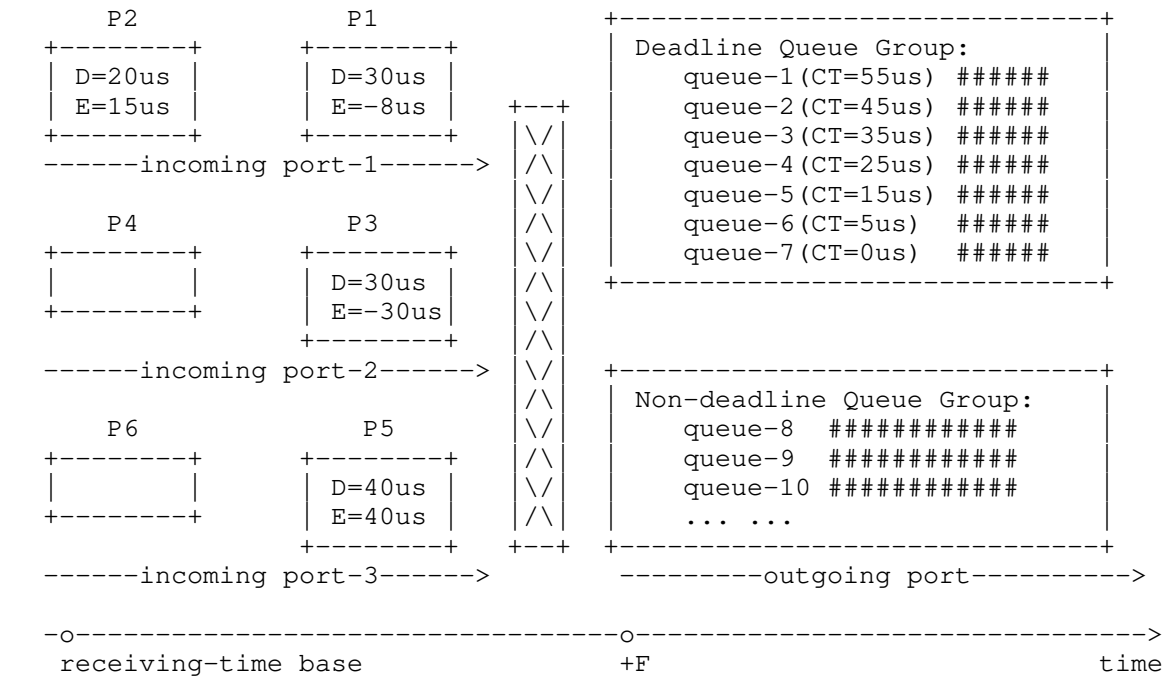


Figure 2: Time Sensitive Packets Buffered to Deadline Queue

As shown in Figure 2, the node successively receives six packets from three incoming ports, among which packet 1, 2, 3 and 5 have corresponding deadline information, while packet 4 and 6 are traditional packets. These packets need to be forwarded to the same outgoing port according to the forwarding table entries. It is assumed that they arrive at the line card where the outgoing port is located at almost the same time after the forwarding delay in the node (F = 5us). At this time, the queue status of the outgoing port is shown in the figure. Then:

* The allowable queueing delay (Q) of packet 1 in the node is 30 - 8 - 5 = 17us, and it will be put into the deadline queue-5 (its CT is 15us), meeting the condition that Q is in the range [15, 25).

* The allowable queueing delay (Q) of packet 2 in the node is 20 + 15 - 5 = 30us, and it will be put into the deadline queue-4 (its CT is 25us), meeting the condition that Q is in the range [25, 35).

* The allowable queueing delay (Q) of packet 3 in the node is 30 - 30 - 5 = -5us, and it will be modified to the minimum positive value 5 us then put into the deadline queue-6 (its CT is 5us), meeting the condition that Q is in the range [5, 15).

* The allowable queueing delay (Q) of packet 5 in the node is 40 + 40 - 5 = 75us, and it will be modified to the MAX_CT (60 us) then put into the deadline queue-1 (its CT is 55us), meeting the condition that Q is in the range [55, 65).

* Packets 4 and 6 will be put into the non-deadline queue in the traditional way.

4.1. Alternate Queue Allocation Rules

One option may further divide a deadline queue into multiple sub-queues, each for a typical planned residence time (D) such as 10us, 20us, ..., 80us. That is, packets with different planned residence time are inserted into different sub-queues and protected. In this way, for two packets with the same allowable queueing delay (Q) but different planned residence time, we can decide to firstly schedule the packet with the smallest planned residence time. This is beneficial because when packets with smaller planned residence time facing larger interference delay, it is difficult to have space for latency compensation on downstream nodes, while packets with larger planned residence time have larger space for latency compensation. However, this option may cause in-time traffic with high priority (small planned residence time) to preempt the sending time of on-time traffic with low priority (large planned residence time).

Another option may further divide a deadline queue into multiple sub-queues, each for a typical residence time deviation (E) such as -60, -50us, -40us, ..., 0, PV(positive value). That is, packets with different existing accumulated residence time deviation are inserted into different sub-queues and protected. In this way, for two packets with the same allowable queueing delay (Q) but different existing accumulated residence time, we can decide to firstly schedule the packet with smallest existing accumulated residence time deviation.

An implementation can also maintain sub-queues with different sub-priorities. Packets are mapped to different sub-priorities according to their planned residence time (D) and accumulated residence time deviation (E), and placed in corresponding sub-queues. The basic principle is to schedule packets with smaller accumulated residence time deviation (E) first. If the accumulated residence time deviation (E) is equal, then schedule packets with smaller planned residence time (D).

5. Buffer Size Design

We assume that the service rate of the deadline scheduler can be the link rate (termed as C), so the buffer size of each deadline queue is AT * C - M, where M is the maximum size of the packet with low priority.

6. Schedulability Condition for Deadline Mechanism

In this section, we first discuss the schedulability condition of in-time scheduling mode. Then analyze the condition of the on-time mode.

Consider that a node is traversed by multiple paths, and each path may carry one or more service flows. For each service flow, it follows the corresponding constraint function, such as the leaky bucket arrival curve A(t) = b + r*t. Generally there are two mechanisms to enforce that traffic of a service entering the deadline scheduler conforms to the given traffic constraint function. The first such mechanism is a traffic policer at the entrance of the network which rejects traffic exceeding the constraint, and the second mechanism is a traffic re-shaper on the intermediate node which temporarily buffers packets to make it conform to the constraint.

Suppose that all nodes along each path have reserved corresponding bandwidth resources according to all service flows carried by that path, then we can classify all service flows forwarded by a node based on different planned residence times. For example, for the class of planned residence time d_i, the corresponding accumulated constraint function is A_i(t). Let d_i < d_(i+1), then the schedulability condition for deadline in-time mode is:.

* A_1(t-d_1) + sum{A_i(t+AT-d_i) for all i>=2} <= C*t

where AT is the authorization time of each deadline queue, C is service rate of the deadline scheduler.

The proof is similar with that in [RPQ], except that the step length is fine-grained by TI when the queue rotates. Figure 3 below gives a rough explanation.

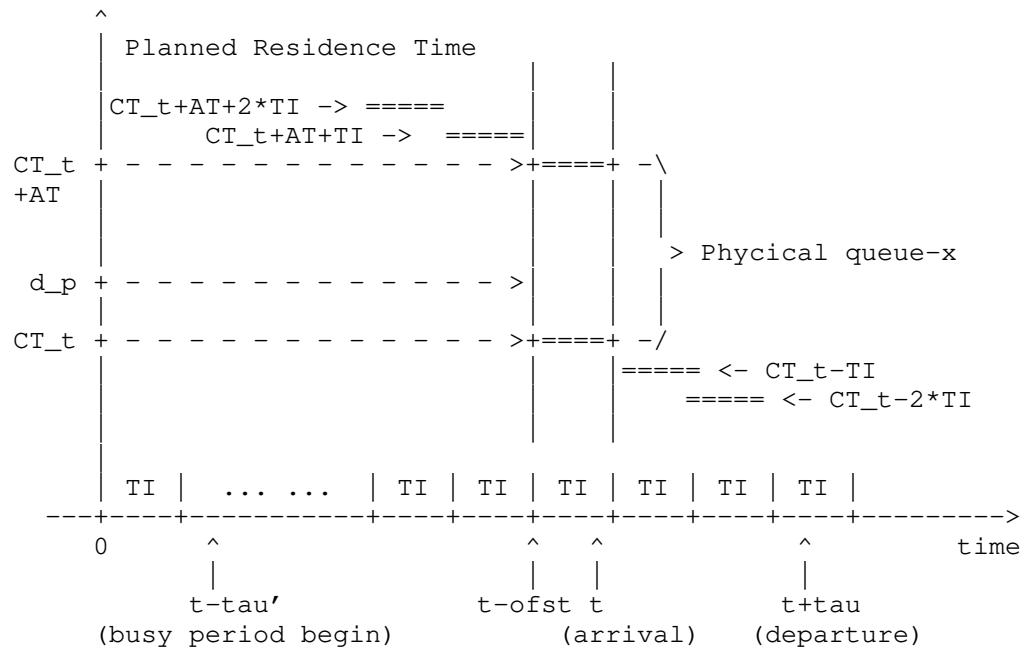


Figure 3: Deadline In-time Scheduling

Suppose that the observed packet, with planned residence time d_p, arrives at the scheduler at time t and leaves the scheduler at time t+tau. It will be inserted to physical queue-x with count-down time CT_t at the current timer interval TI with starting time t-ofst and end time t-ofst+TI. According to the above packet queueing rules, we have CT_t <= d_p < CT_t+AT. Also suppose that t-tau' is the beginning of the busy period closest to t. Then, we can get the amount of packets within time interval [t-tau', t+tau] that must be scheduled before the observed packet. In detailed:

* for all service i with planned residence time d_i meeting CT_t <= d_i < CT_t+AT, the workload is sum{A_i[t-tau', t]}.

  Explanation: since the packets with priority d_i in the range [CT_t, CT_t+AT) at time t will be sent before the observed packet, the packets with the same priority d_i before time t will become more urgent at time t, and must also be sent before the observed packet.

* for all service i with planned residence time d_i meeting d_i >= CT_t+AT, the workload is sum{A_i[t-tau', t-ofst+TI-(d_i-CT_t-AT+TI)]}.

Explanation: although the packets with priority d_i larger than CT_t+AT at time t will be sent after the observed packet, but the packets with the same priority d_i before time t, at time t-ofst+TI-(d_i-CT_t-AT+TI), will become more urgent at time t, and must be sent before the observed packet.

* for all service i with planned residence time d_i meeting d_i < CT_t, the workload is sum{A_i[t-tau', t+(CT_t-d_i)]}.

Explanation: the packets with priority d_i less than CT_t at time t will certainly be sent before the observed packet, at a future time t+(CT_t-d_i) the packets with the same priority d_i will still be urgent than the observed packet (even the observed packet also become urgent), and must be sent before the observed packet.

* then deduct the traffic that has been sent during the busy period, i.e., C*(tau+tau').

Let tau as d_p, and remember that CT_t <= d_p, the above workload is less than

```
sum{A_i(tau'+CT_t+AT-d_i) for all d_i >= CT_t} +
sum{A_i(tau'+CT_t-d_i) for all d_i < CT_t} - C*(tau'+d_p)
```

It is further less than

```
sum{A_i(tau'+d_p+AT-d_i) for all d_i >= d_2} + A_1(tau'+d_p-d_1) -
C*(tau'+d_p)
```

Then, denote x as tau'+d_p, we have

```
sum{A_i(x+AT-d_i) for all d_i >= d_2} + A_1(x-d_1) - C*(x)
```

Let the above workload less than zero, then we get the condition.

It should be noted that for a class i, its planned residence time d_i is actually contributed by its own flows and all the classes with lower planned residence time. Thus, based on the above schedulability conditions, and knowing the traffic constraint functions of all classes, we can then give the guidance to allocate appropriate (i.e., not arbitrary) planned residence time for each class. In brief, we can choose the traffic arrival constraint function according to the preset planned residence time, or we can choose the planned residence time according to the preset traffic arrival constraint function.

Deadline in-time mode is a variant of EDF scheduler with work-conserving characteristic and has several aggregated queues. It needs to rely on the known traffic arrival curve to obtain the sufficient and necessary schedulability conditions. In traffic aggregation case, the arrival traffic faced by the intermidiate nodes may be irregular due to work-conserving scheduling.

* One option may choose to implement re-shaping per flow or interleaved regulator (IR) introduced by [UBS] placed before the deadline schedulre on the intermediate node. It is not recommended to maintain re-shapers per flow in a large-scale network. As described in [IR-Theory], re-shaper per flow or interleaved regulator does not increate worst-case latency bounds.

* Another option is that the deadline queue based on latency compensation has essentially played the role of re-shaper queue. Assume that an in-time burst A_i released late in the network quickly reaches the intermediate node, and catches up with the traffic released early in the network, resulting in the actual arrival curve of A_i exceeding the constraints. However, the excess traffic can be identified by the scheduler, which has a greater residence time deviation and is placed in a queue with a larger CT value, without causing interference to class i. Although the excess traffic may be ranked before class j (j>i) after latency compensation, considering that the essence of latency compensation is to align to the on-time mode, that is, even if the excess traffic is sent in the on-time mode, this part of traffic naturally needs to be sent before class j traffic.

The test of schedulability conditions needs to be based on the whole network view. When we need to add new traffic to the network, we need to consider which nodes the related path will pass through, and then check in turn whether these nodes still meet the schedulability conditions after adding new traffic.

6.1. Schedulability Condition Analysis for On-time Mode

Compared with the in-time mode, on-time mode is non-work-conserving, which can be considered as the combination of damper and EDF scheduler. The intuitive understanding of the on-time mode is that if the headends of multiple paths (they converge at the same intermediate node) apply admission control strictly according to the service bandwidth when releasing traffic to the network, and successfully reserve the corresponding bandwidth resources for these paths in the control plane, then the on-time forwarding behavior on a path controls the interval between early and late release of traffic on that path, but not lead to an increase in the bandwidth occupied by that path. Therefore, the on-time mode does not cause the arrival

curve to exceed the expected traffic constraint function.

If any packets with on-time mode can always be sent before or at their deadline, e.g, place packets to deadline queues according to CT <= Q-k*AT < CT+AT, the above schedulability condition can also be applied to the on-time mode, that is, if the aggregate traffic for the specific planned residence time reaches the maximum limit allowed by the conditions, it will achieve passive on-time behavior. If the aggregate traffic is lower than the limit, the deadline on-time scheduler will also achieve active on-time behavior by applying the built-in damper. Since the built-in dumper has shaped the on-time service flow, there is also no need to re-shape the on-time service flow by additional shapers in front of deadline scheduler.

However, this is not always the case. Since the on-time mode explicitly introduces the hold time, the actual departure time of the packet may be after the deadline. For example, a set of packets with class of d_60 (i.e., the planned residence time is 60 us) arrives at the deadline scheduler at time T, assuming that these packets require a total transmission time of 10us. Then after 50 us, another set of packets with class of d_10 (i.e., the planned residence time is 10 us) arrives, also assuming a total transmission time of 10us. These two batches of packets will start to be sent at the same time, which will inevitably lead to some packets not leaving before their deadline. Although the delayed packets can be accelerated on the downstream nodes through latency compensation, it still bring risks to end-to-end QoS target.

On the one hand, hold time is used to avoid jitter, but on the other hand, it is also a waste of transmission opportunities.

This means that implementing more relaxed on time scheduling on intermediate nodes will be more beneficial to end-to-end QoS target. For example, for CT <= Q-k*AT < CT+AT, one can config a large k (>1) on the intermediate nodes for loose on-time scheduling, while config a small k (<1) on the egress for strict on-time scheduling. A node needs to identify its role as an intermediate node or egress node for a specific packet and provide an appropriate k value. Note that even if all intermediate nodes apply loose on-time scheduling, as long as the egress node applies strict on-time scheduling, the jitter target can still be achieved.

7. Traffic Regulation and Orchestrating

On the ingress PE node, traffic regulation must be performed on the incoming port, so that the service traffic does not exceed its constraint funciton, such as leacky bucket A_i(t) = b_i + r_i*t for service flow i.

Then, on the ingress PE node, traffic is orchestrated into different deadline queues of the outgoing port. Multiple continuous packets of the specific service flow are stored in the deadline queue with corresponding remaining time according to the planned residence time of the service flow. Note that these packets are not stored in the same queue over time. The amount of bits that can be stored in one queue for service i is equal to r_i * AT, however, at least one whole packet shall be loaded (i.e., there may be overprivisioning per AT in the case of large packet size and small AT value). For example, if the allowable queueing delay (Q) is 20us, then within the current authorization time, the first sequence of the packets will be put into the current deadline queue with [CT, CT+AT) range that can cover 20us, until the reserved bandwidth limit is reached; Then, within the next authorization time, the next sequence of packets will be put into the queue with [CT, CT+AT) range that can cover 20us, until the reserved bandwidth limit is reached; and so on, until the total service bits are loaded.

Figure 4 depicts an example of deadline based traffic orchestrated on the ingress PE node. It is assumed that the packets loaded in each authorization time do not exceed the reserved bandwidth of the service.

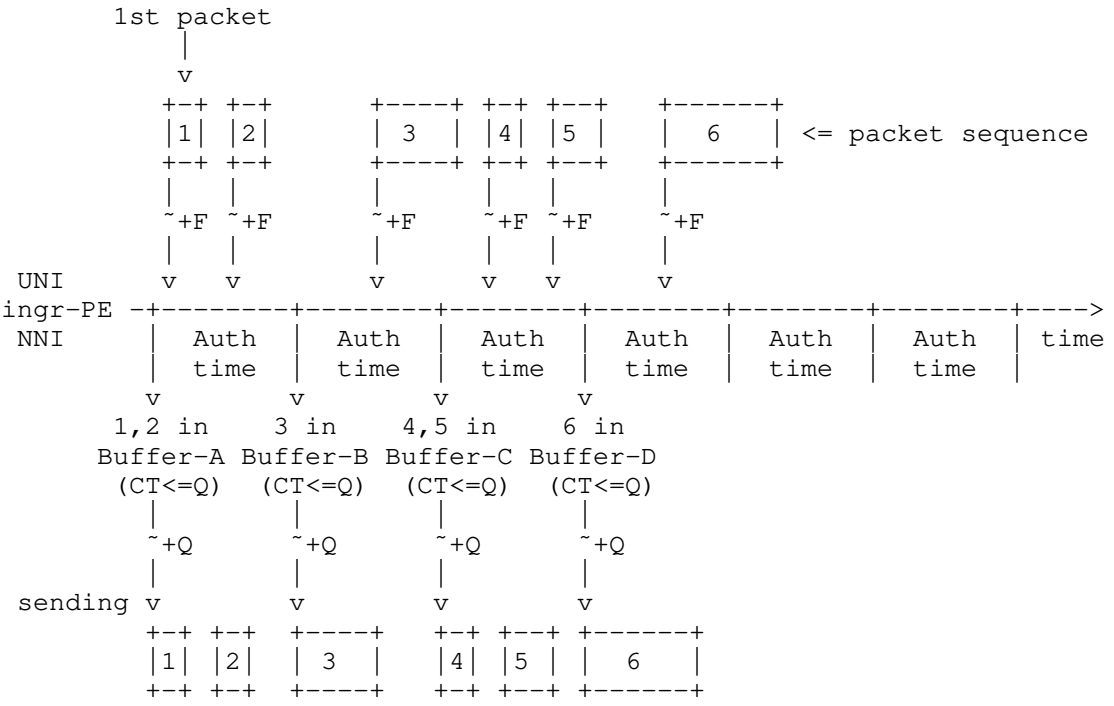


Figure 4: Deadline Based Packets Orchestrating

8. Latency Compensation Considerations

As mentioned above, the calculation of allowable queueing delay (Q) reflects the meaning of latency compensation. It can be used for in-time services to distinguish between emergency flows and non emergency flows (compared with traditional strict priority scheduling), and also for on-time services to strictly control delay jitter. In both cases, packets have a tolerable maximum queueing delay in the queueing subsystem. The traditional EDF scheduling can also benefit from this latency compensation.

Suppose that two packets, P1, P2, are generated instantaneously from a specific in-time service flow at the same source, and the two packets have the same planned residence time. Packet-1 may face less interference delay than P2 in their process of forwarding. When they arrive at an intermediate node successively, P2 will have less allowable queueing delay (Q) than P1 to try to stay close to P1 again. It should be noted that to compary who is ealier is based on queue's CT and packet's Q, according to the above queueing rule (CT <= Q < CT+AT), and the CT of the queue is not changed in real-time, but gradually with the decreasing step TI. It is possible to get an unexpected comparision result.

As shown in Figure 5, P1 and P2 are two packets belonging to the same burst. The arrival time when they are received on the current scheduler is shown in the figure. Suppose that the AT of the deadline queue is 10us, the decreasing step TI is 1us, and the transmission time of each packet is 0.01us. Also suppose that the Q values of two adjacent packets P1 and P2 are 40us and 39us respectively, and they are both received in the window from T0 to T0+1us. P1 will enter queue-B with CT range [40, 50), while P2 will enter queue-A with CT range [30, 40). This means that P2 will be scheduled before P1, resulting in disorder.

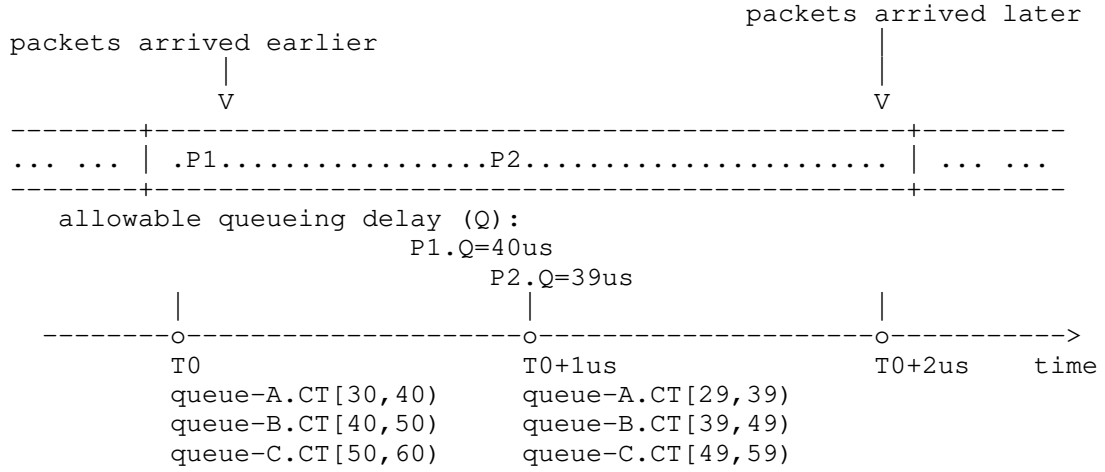


Figure 5: Packets queueing based on Latency Compensation

DetNet architecture [RFC8655] provides Packet Ordering Function (POF), that can be used to solve the above disorder problem caused by the latency compensation.

Alternatively, if the POF is not enabled, we can also maintain states for service flows to record the last queueing information to address this issue.

For example, one ore more OGOs (order guarantee object) are maintained per flow (or per incoming port, planned residence time, incoming port plus planned residence time, etc), on each outgoing port. An OGO records the queue information which is the queue that all the packets mapped to this OGO was inserted recently. For simplicity, a count-down time (CT), which is copied from the recent inserted deadline queue, can be recorded in OGO. Note that the CT of OGO needs to decrease synchronously with that of other deadline queues, with the same decreasing step TI. If the CT of OGO decreases to 0, it will remain at 0.

When a packet arrives at the deadline scheduler at the outgoing port, it is first mapped to its OGO, and get the CT of OGO, termed as OGO.CT. Then, according to the above queueing rule (CT <= Q-k*AT < CT+AT), get the CT of a preliminarily selected queue, termed as preliminary CT.

> Let Q' is MAX{OGO.CT, preliminary CT}, and put the packet in the target queue according to CT <= Q' < CT+AT
>
> Update the value of OGO.CT to the CT of target queue.

The intuitive meaning of the above operation is that for two packets belonging to the same OGO (e.g, per flow), the packet received later should not be queued before the packet received first. This also implies that the main purpose of latency compensation is to compensate the interference delay between pakckets of different flows, rather than compensating the interference delay between packets of the same flow.

9. Compatibility Considerations

For a particular path, if only some nodes in the path upgrade support the deadline mechanism defined in this document, the end-to-end deterministic delay/jitter target will only be partially achieved. Those legacy devices may adopt the existing priority based scheduling mechanism, and ignore the possible deadline information in the packet, thus the delay intra node produced by them cannot be perceived by the adjacent upgraded node. The more upgraded nodes included in the path, the closer to the delay/jitter target. Although, the legacy devices may not support the dataplane mechanism described in this document, but they can be freely programmed (such as P4 language) to measure and insert the deadline information into packets, in this case the achievement of delay/jitter target will be

more perfect.

Only a few key nodes are upgraded to support deadline mechanism, which is low-cost, but can meet a service with relatively loose time requirements. Figure 6 shows an example of upgrading only several network border nodes. In the figure, only R1, R2, R3 and R4 are upgraded to support deadline mechanism. A deterministic path across domain 1, 2, and 3 is established, which contains nodes R1, R2, R3, and R4, as well as explicit nodes in each domain. Domain 1, 2 and 3 use the traditional strict priority based forwarding mechanism. The encoding of the packet sent by R1 includes the planned residence time and the accumulated residence time deviation. Especially, DS filed in IP header ([RFC2474]) are also set to appropriate values. The basic principle of setting is that the less the planned residence time, the higher the priority. However, in order to avoid the interference of non deterministic flow to deterministic flow, the priority of deterministic flow should be set as high as possible.

The delay analysis based on strict priority in each domain can be found in [SP-LATENCY], which gives the formula to evaluate the worst-case delay of each hop during the resource reservation procedure. The worst-case delay depends on the number of hops and the burst size of interference flows that may be faced on each hop. [EF-FIFO] also shows that, for FIFO packet scheduling be used to support the EF (expedited forwarding) per-hop behavior (PHB), if the network utilization level alpha < 1/(H-1), the worst-case delay bound is inversely proportional to 1-alpha*(H-1), where H is the number of hops in the longest path of the network. Although the deadline in-time scheduling, the SP scheduling and EF FIFO scheduling are all work-conserving, the deadline in-time scheduling can further distinguish between emergency and non emergency according to deadline information other than traffic class. Therefore, when analyzing the latency of the in-time mode, the latency is not evaluated just according to the order in which the packets arrive at the scheduler, but also according to the deadline of the packets. An intuitive phenomenon is that if a packet unfortunately faces more interference delays at the upstream nodes, it will become more urgent at the downstream node, and will not always be unfortunate. This operation of dynamically modifying the key fields, e.g, the existing acumulated residence time deviation (E), of the packet can avoid always overestimating worst-case latency on all hops. According to schedulability condition, in-time mode can be pessimistic that its worst-case latancy bounds is the same as on-time mode.

When the boundary node (e.g, R2) receives the deterministic traffic, it will decide whether to speed up or hold according to the existing accumulated residence time deviation information carried in the packet. The in-time traffic is always sent as soon as possible,

especially when the residence time deviation of the packet is negative. The on-time traffic always controls the sending time so that the average planned residence time is followed, especially when the residence time deviation of the packet is positive. For a specific deterministic flow, if it experiences too much latency in the SP domain (due to unreasonable setting of traffic class and the inability to distinguish between deterministic and non deterministic flows), even if the boundary node accelerates the transmission, it may not be able to achieve the target of low E2E latency. If the traffic experiences less latency within the SP domain, on-time mode will work to achieve the end-to-end jitter target.

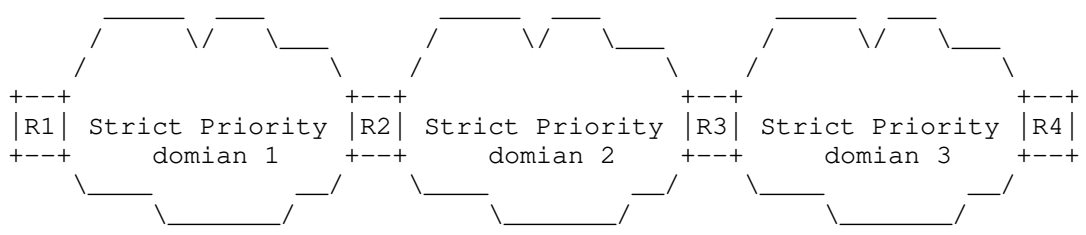


Figure 6: Example of partial upgrade

10. Deployment Considerations

According to the above schedulability conditions, the planned residence time that can be provided in the network is related to the entire deployed service flows. Each planned residence time can be considered as a delay resource, and the smaller the residence time, the more valuable it is. The operator needs to match the corresponding planned residence time for each service. It is not recommended to allocate a large planned residence time for the on-time service, which will consume a large amount of buffer and is not beneficial to target jitter. However, the in-time service may also not want to be assigned to a larger planned residence time, that may be determined by its SLA that requires a smaller target end-to-end latency. Thus it is necessary to find a balance to meet more service requirements as much as possible.

We believe that the planned residence time specified for the service flows is completely determined by the network, not by the user. Therefore, it is impossible to carry the planned residence time beyond the network capacity in the service packets. However, there may be some services that expect a large residence time, in this case

it is recommended to hold them at the border nodes, but not on the intermediate node. For this purpose, the MAX_CT configured on the border nodes can be much larger than the MAX_CT configured on the intermediate nodes, or the hierarchical deadline queues can be configured on the border nodes.

As mentioned above, loose on-time scheduling or in-time scheduling can be applied in the network, and the jitter is removed at the edge node according to strict on-time scheduling.

11. Benefits

The mechanism described in this document has the following benefits:

* Time synchronization is not required between network nodes. Each node can flexibly set the authorization time length of the deadline queue according to its own outgoing port bandwidth.

* Packet multiplexing based, it is an enhancement of PQ scheduling algorithm, friendly to the upgrade of packet switching network. All nodes in the network can independently use cycle timers with different timeout intervals to rotate the deadline queues.

* The packet can control its expected residence time in the node. A single set of deadline queues supports multiple levels of residence time.

* For in-time mode, the end-to-end delay is H*(F~D), jitter is H*Q; For on-time mode, the end-to-end delay is H*D, jitter is a just single authorization time.

12. IANA Considerations

There is no IANA requestion for this document.

13. Security Considerations

TBD

14. Acknowledgements

TBD

15. References

15.1. Normative References

[I-D.peng-6man-deadline-option]
Peng, S., Tan, B., and P. Liu, "Deadline Option", Work in Progress, Internet-Draft, draft-peng-6man-deadline-option-01, 11 July 2022, <https://www.ietf.org/archive/id/draft-peng-6man-deadline-option-01.txt>.

[I-D.stein-srtsn]
Stein, Y. J., "Segment Routed Time Sensitive Networking", Work in Progress, Internet-Draft, draft-stein-srtsn-01, 29 August 2021, <https://www.ietf.org/archive/id/draft-stein-srtsn-01.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC9320] Finn, N., Le Boudec, J.-Y., Mohammadpour, E., Zhang, J., and B. Varga, "Deterministic Networking (DetNet) Bounded Latency", RFC 9320, DOI 10.17487/RFC9320, November 2022, <https://www.rfc-editor.org/info/rfc9320>.

15.2. Informative References

[EF-FIFO] "Fundamental Trade-Offs in Aggregate Packet Scheduling", 2001, <https://ieeexplore.ieee.org/document/992892>.

[IR-Theory]
"A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators", 2018, <https://ieeexplore.ieee.org/document/8519761>.

[RPQ] "Exact Admission Control for Networks with a Bounded Delay Service", 1996, <https://ieeexplore.ieee.org/document/556345/>.

[SP-LATENCY] "Guaranteed Latency with SP", 2020, <https://ieeexplore.ieee.org/document/9249224>.

[UBS] "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks", 2016, <https://ieeexplore.ieee.org/abstract/document/7557870>.

Authors' Addresses

Shaofu Peng
ZTE Corporation
China
Email: peng.shaofu@zte.com.cn

Bin Tan
ZTE Corporation
China
Email: tan.bin@zte.com.cn

Peng Liu
China Mobile
China
Email: liupengyjy@chinamobile.com

DetNet Working Group
Internet-Draft
Intended status: Standards Track
Expires: 10 May 2023

X. Song
Q. Xiong
ZTE Corp.

6 November 2022

MPLS Sub-Stack Encapsulation for Deterministic Latency Action
draft-sx-detnet-mpls-queue-03

Abstract

This document specifies formats and principals for the MPLS header which contains the Deterministic Latency Action (DLA) information, designed for use over a DetNet network with MPLS data plane. It enables guaranteed delay support and makes scheduling decisions for time-sensitive service running on DetNet nodes that operate within a constrained network domain.

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 10 May 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

As specified in [RFC8655] and [RFC8938], Deterministic Networking (DetNet) operates at the IP layer and delivers service with low data loss rates and bounded latency guarantee within a network domain.

As defined in [RFC8964], the DetNet MPLS data plane provides a foundation of building blocks to enable PREOF (Packet Replication, Elimination and Ordering Functions (PREOF)) functions to DetNet service and forwarding sub-layer. The DetNet service sub-layer includes a DetNet Control Word (d-CW), service label (S-Label), an aggregation label (A-Label) in special case of S-Label used for aggregation. The DetNet forwarding sub-layer supports one or more forwarding labels (F-Labels) used to forward a DetNet flow over MPLS domains. The DetNet forwarding sub-layer provides corresponding forwarding assurance with IETF existing functions using resource allocations and explicit routes. But these functions can not provide the deterministic latency (including bounded latency, low packet loss and in-order delivery) assurance.

To support time-sensitive service with ultra-low loss rates and deterministic latency, it is required to apply feasible scheduling mechanisms to specific applications for deterministic networking. As described in [I-D.ietf-detnet-bounded-latency], the end-to-end bounded latency is considered as the sum of non-queuing and queuing delay bounds along with the queuing mechanisms. The value for non-queuing delay bounds (which consist of packet output delay, link delay, frame preemption delay and processing delay) is relative with the physical capability of on-used networks and can be considered to be stable. The unstable latency delay bounds are mainly from queuing delay and regulation delay. The regulation delay is mainly from

regulation policy. To simplify the question this draft assumes there is no regulation policy. So the question is left to address the selection for queuing mechanisms and queuing delay information encapsulation in data plane.

The queuing mechanisms, as mentioned in [I-D.ietf-detnet-bounded-latency] and [RFC8655], which include Time Aware Shaping IEEE802.1Qbv, Asynchronous Traffic Shaping IEEE802.1Qcr, cyclic-scheduling queuing mechanism proposed in IEEE802.1Qch. In terms of delay guarantee, to select the right scheduling/queuing mechanism applied to a specific application is required.

Based on the existing DetNet MPLS encapsulations and mechanisms [RFC8964], the draft defines the encoding format for Deterministic Latency Action (DLA) carried in MPLS sub-stack.

2. Conventions

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Terminology

Refer to [RFC8655], [RFC8964], [I-D.jags-mpls-mna-hdr] and [I-D.ietf-detnet-bounded-latency] for the key terms used in this document.

Deterministic Latency (DL):the bound of network latency and delay variation between two DetNet endpoints. It may includes parameters such as bounded latency, bounded delay variation, etc.

Deterministic Latency Action (DLA): used to indicate deterministic latency actions for MPLS Sub-Stack.

3. DetNet queue encapsulation with MPLS data plane

3.1. Queuing delay

[RFC8655] provides the architecture for deterministic networking (DetNet) which enables the service delivery of DetNet flows with extremely low packet loss rates and deterministic latency. The forwarding sub-layer provides corresponding forwarding assurance but can not provide the deterministic latency (including bounded latency, low packet loss and in-order delivery). As described at [I-D.ietf-detnet-bounded-latency], the end-to-end bounded latency for one DetNet flow is the sum of delay bound of non-queuing and queuing processing latency. The delay bound for non-queuing processing may include output delay, link delay, frame preemption delay, and processing delay, the delay bound for queuing processing may include regulator delay, queuing delay. It is assumed that the delay of non-queuing processing is fixed or be ignorable, the delay of queuing processing is variable. To realize the guarantee of bounded latency service it is important to select right queuing methodology applied to specific applications and carry necessary queuing delay information for computation of end-to-end latency.

The DetNet data plane encapsulation in transport network with MPLS data plane is specified in [RFC8964]. This document provides additional encapsulation for the DLA in MPLS data plane.

3.2. DetNet MPLS Encapsulation with Delay Option

The DetNet MPLS header follows [RFC8964], as showed the below figure 1, the SP-Lable (SPL) is added to indicate Deterministic Latency Action (DLA).

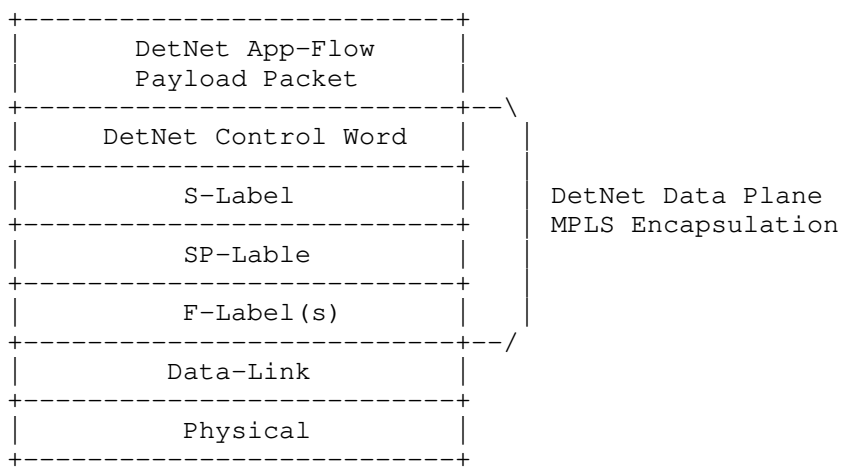


Figure 1: DetNet MPLS Header

The SP-L (Special Purpose-Label) may be B-SPL [RFC9017], new-SPL, extended SPL [RFC9017]. This draft follows the MNA (MPLS Network Action) solution specified in [I-D.jags-mpls-mna-hdr] and [I-D.ietf-mpls-mna-fwk], and uses b-SPL to indicate the presence of the MPLS Network Action Sub-Stack (NASS). The value for the bSPL value is to be assigned by IANA and follows the assignment in [I-D.jags-mpls-mna-hdr]. The SP-Label field is formatted as below figure.

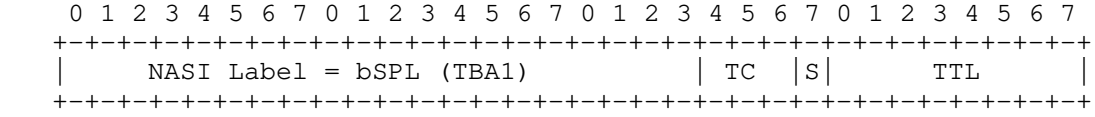


Figure 2: SP-Label Format

NASI Label:

A new bSPL value is to be assigned by IANA. It is used to indicate the presence of the MPLS Network Action Sub-Stack (NASS). The assignment for this field value refers to [I-D.jags-mpls-mna-hdr].

The MPLS sub-stack encoding format for DetNet Latency Action (DLA) is showed as figure 3. The format provides DetNet Latency Network Action Indicator (NAI) indicates the specific DLA. Its detailed information is carried in Ancillary Data.

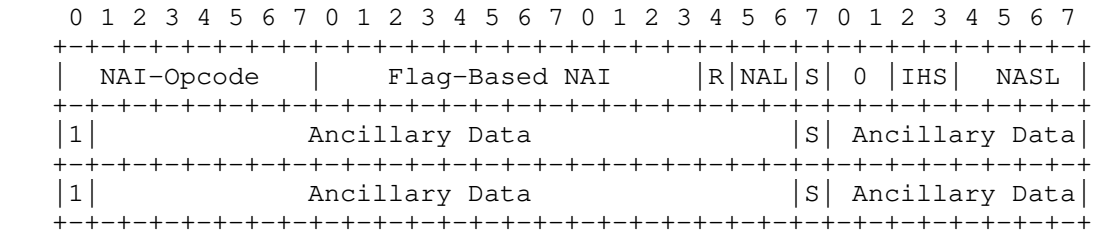


Figure 3: MPLS LSE Format for DLA

NAI-Opcode:

This is the first 8-bit value in the Label Field. The value is used to indicate DLA and to be assigned by IANA. It ranges from 0 to 255.

Flag-Based NAIs:

The Flag-Based NAIs field carries flags for DetNet Queuing mechanisms.

R bit:

R bit in the TC field is Reserved for future use.

NAL (Network Action Length):

The 2-bit field in TC is used to carry the number of additional LSEs used to carry the Ancillary Data for the Network Action. The NAL value for DetNet MUST set to 2.

S bit:

Indicator for bottom of MPLS stack.

P,H (Post-Stack Network Action Presence and Post-Stack Hop-By-hop processing Indicator) bit:

This is 2-bit flag, where "P" bit indicates the presence of Post-Stack Network Actions and "H" bit indicates the presence of Post-Stack Hop-By-Hop and/or Select processing scope options. While encoding the Post-Stack NAs, the HBH/Select scope NAs MUST be encoded first (closer to the BOS) and then I2E. The DetNet Queuing Option is proposed to use in-stack encoding, the P,H field is set to 0.

IHS (I2E, HBH, and Select Scope):

This 2-bit value indicates the scope of In-Stack NAIs. DetNet Queuing Option is considered to be processed by HBH, so the value is set to 01 refers to table 1 in [I-D.jags-mpls-mna-hdr].

NASL (Network Action Sub-Stack Length):

This is a 4-bit field in the TTL. This indicates the total length of the current NASS.

The first bit in the Label field of the second and third LSE MUST be set to "1". As specified in [I-D.jags-mpls-mna-hdr] this is to prevent aliasing the label field with other bSPLs on the legacy routers.

Ancillary Data:

The 19-bit Label field and 4-bit TC field and 8-bit TTL field (except S bit) in the additional LSEs are used to carry the Ancillary Data for specific DetNet queuing delay information.

The Flag-Based NAIs field is designed as follow:

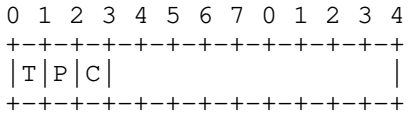


Figure 4: Flag-based NAIs

T Flag: 1 bit, TAS (Time Aware Shaping) [IIEEE802.1Qbv] queuing indicator. If the flag is set 1, the TAS is used for the DetNet flows, the can be correlated with the special label encapsulated in figure 2, or be mapped with F-Label (s) or S-Label showed in figure 1.

P Flag: 1 bit, Credit-Based Shaper [IEEE802.1q-2014] queuing indicator.

C Flag: 1 bit, CQF [IEEE802.1Qch] queuing indicator.

Note: For one specific DetNet flow, there is one or more choices for queuing mechanisms selection, the queuing mechanisms can be used respectively or combined with each other.

If the flag field is set 1 it will associate its AD (Ancillary Data) for specific queuing delay information. The encapsulation for AD is showed as TLV format.

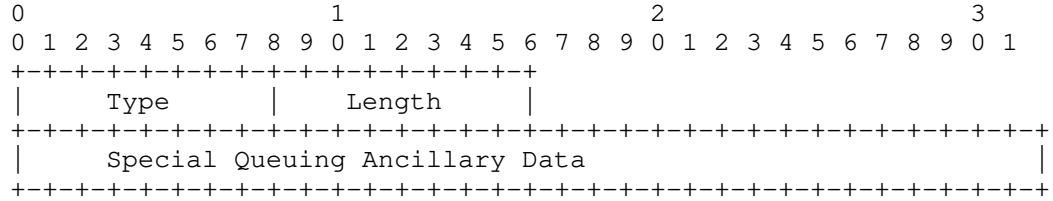


Figure 5: Ancillary data for queuing delay

The encapsulation for specific queuing delay information follows the requirements and framework of MPLS MNA discussed in MPLS WG.

The encapsulation for Special Queuing Ancillary Data field is specific for queuing mechanisms. For example, the deadline queuing may carry deadline delay information, refer to [I-D.peng-6man-deadline-option].The cyclic-scheduling queuing may carry cycle ID related information, refer to

[I-D.dang-queuing-with-multiple-cyclic-buffers]. The detailed format depends on the discussion on the corresponding drafts and left for FFS.

4. IANA Considerations

This document describes a new IANA-managed registry to identify DetNet application processing. The registration procedure is "IETF Review" [RFC8126]. The registry name is "NAI-Opcode" and assigned for DLA Indicator, as defined in Table 1.

Value	Description	Reference
Unassigned	NAI-Opcode	this document

Table 1: DLA Indicator

This document describes a new IANA-managed registry to identify DetNet MPLS Queue Flags Bits. The registration procedure is "IETF Review" [RFC8126]. The registry name is "Flag-based-NAIs". There is 12-bit Flag field, as defined in Table 2.

Bit Position	Symbol	Description	Reference
0	T	TAS Flag	this document
1	P	CBS Flag	this document
2	C	CQF Flag	this document
3-11	Unassigned	Unassigned	this document

Table 2: Flags for DLA Indicator

5. Security Considerations

Security considerations for DetNet are covered in the DetNet Architecture RFC8655 and DetNet Security Considerations [RFC9055]. MPLS security considerations are covered in [RFC8964], [RFC3031], [RFC3032]. These security considerations also apply to this document.

6. Acknowledgements

The authors would like to acknowledge Shaofu Peng for his thorough review and very helpful comments.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC9017] Andersson, L., Kompella, K., and A. Farrel, "Special-Purpose Label Terminology", RFC 9017, DOI 10.17487/RFC9017, April 2021, <https://www.rfc-editor.org/info/rfc9017>.

7.2. 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.ietf-detnet-bounded-latency]
Finn, N., Boudec, J. L., 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.ietf-mpls-mna-fwk]
Andersson, L., Bryant, S., Bocci, M., and T. Li, "MPLS Network Actions Framework", Work in Progress, Internet-Draft, draft-ietf-mpls-mna-fwk-02, 21 October 2022, <https://www.ietf.org/archive/id/draft-ietf-mpls-mna-fwk-02.txt>.

[I-D.jags-mpls-mna-hdr]
Rajamanickam, J., Gandhi, R., Zigler, R., Song, H., and K. Kompella, "MPLS Network Action (MNA) Header Encodings", Work in Progress, Internet-Draft, draft-jags-mpls-mna-hdr-03, 5 November 2022, <https://www.ietf.org/archive/id/draft-jags-mpls-mna-hdr-03.txt>.

[I-D.peng-6man-deadline-option]
Peng, S., Tan, B., and P. Liu, "Deadline Option", Work in Progress, Internet-Draft, draft-peng-6man-deadline-option-01, 11 July 2022, <https://www.ietf.org/archive/id/draft-peng-6man-deadline-option-01.txt>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[RFC9055] Grossman, E., Ed., Mizrahi, T., and A. Hacker, "Deterministic Networking (DetNet) Security Considerations", RFC 9055, DOI 10.17487/RFC9055, June 2021, <https://www.rfc-editor.org/info/rfc9055>.

Authors' Addresses

Xueyan Song
ZTE Corp.
Nanjing
China
Email: song.xueyan2@zte.com.cn

Quan Xiong
ZTE Corp.
Wuhan
China
Email: xiong.quan@zte.com.cn

DETNET
Internet-Draft
Intended status: Standards Track
Expires: 2 April 2023

Q. Xiong
A. Liu
ZTE Corporation

29 September 2022

DetNet Deterministic Latency Option for IPv6
draft-xiong-detnet-6man-queuing-option-02

Abstract

This document introduces new IPv6 options to identify the Deterministic Latency related information for DetNet flows in IPv6 and SRv6 networks.

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 2 April 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

According to [RFC8655], Deterministic Networking (DetNet) operates at the IP layer and delivers service which provides extremely low data loss rates and bounded latency within a network domain. DetNet data planes has been specified in [RFC8938]. The existing deterministic technologies are facing large-scale number of nodes and long-distance transmission, traffic scheduling, dynamic flows, and other controversial issues in large-scale networks. The enhanced DetNet Data plane is required to support a data plane method of flow identification and packet treatment.
[I-D.liu-detnet-large-scale-requirements] has described the enhancement requirements for DetNet data plane, it is required to support information used by functions ensuring Deterministic Latency.
[I-D.xiong-detnet-large-scale-enhancements] has proposed the overall framework of DetNet enhancements for large-scale deterministic networks. The packet treatment should support new functions to ensure deterministic latency and the identification of deterministic latency related information.

As described in [I-D.ietf-detnet-bounded-latency], the end-to-end bounded latency depends on the value of queuing delay bound along with the queuing mechanisms. Multiple queuing mechanisms can be used to guarantee the bounded latency in DetNet. And many types of queuing mechanisms have been proposed to provide diversified deterministic service for various applications. For example, time-scheduling queuing mechanisms includes the Time Aware Shaping [IIEEE802.1Qbv] and priority-scheduling includes the Credit-Based

Shaper[IEEE802.1Q-2014] with Asynchronous Traffic Shaping[IEEE802.1Qcr]. The cyclic-scheduling queuing mechanism has been proposed in [IEEE802.1Qch] and improved in [I-D.dang-queuing-with-multiple-cyclic-buffers]. The deadline-scheduling queuing mechanism has been proposed in [I-D.stein-srtsn] and improved in [I-D.peng-detnet-deadline-based-forwarding]. The per-flow queuing mechanism includes Guaranteed-Service Integrated service (IntServ) [RFC2212]. The asynchronous queuing mechanism includes the Asynchronous Deterministic Networking (ADN) as per [I-D.joung-detnet-asynch-detnet-framework]. The functions such as the queuing mechanisms should be provided for enhanced DetNet to ensure the deterministic latency. So it is required to carry deterministic latency related information such as queuing parameters in data plane so as to make appropriate packet forwarding and scheduling decisions to meet the time bounds. The DetNet forwarding nodes along the path can apply the function and the deterministic latency related information carried in the packet to achieve the end-to-end bounded latency.

This document introduces new IPv6 options to identify the Deterministic Latency related information for DetNet flows in IPv6 and SRv6 networks.

2. Conventions used in this document

2.1. Terminology

The terminology is defined as [RFC8655].

2.2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. The DetNet Options

This document defines new IPv6 options for DetNet to signal Deterministic Latency related information to the DetNet layers. The format of the options follow the generic definition in section 4.2 of [RFC8200]. The option may be placed either in an HbH or a DoH EH.

3.1. The Deterministic Latency information Option

The DetNet Deterministic Latency Information Option helps to discriminate the types of mechanisms and specify the related parameters.

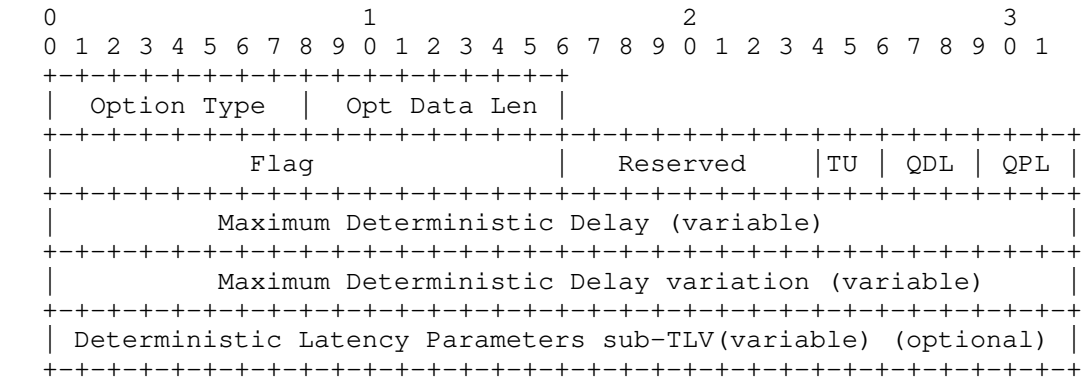


Figure 1: Deterministic Latency Option Format

Option Type: 8-bit identifier of the type of option. Value TBD1 by IANA; the highest-order 3 bits of the field is 001 to skip over this option and continue processing the header if the processing IPv6 node does not recognize the Option Type and to permit the Option Data to be changed en route to the packet's final destination.

Opt Data Len: 8-bit unsigned integer. Length of the Option Data field of this option, in octets. It is set to 12.

Flag(16 bits): indicates the type of Deterministic Latency function. The flag is defined in section 3.1.1.

TU(2 bits): indicates the time units for the Deterministic Delay.

ODL(3 bits): Length of the Deterministic Delay field as an unsigned 3-bit integer. The length of Deterministic Delay variation field is same with Deterministic Delay field.

OPL(3 bits): Length of the Deterministic Latency Parameters field as an unsigned 3-bit integer.

Maximum Deterministic Delay (variable): indicates the Maximum queuing delay. The value is various when the Queuing type is different.

Maximum Deterministic Delay variation (variable): indicates the Maximum Deterministic delay variation. The value is various when the flag is different.

Deterministic Latency Parameters Sub-TLVs (variable): it is optional and provides additional information for a node to forward a DetNet flow. The Sub-TLVs has been defined in section 3.1.2.

3.1.1. Flag

The types of Deterministic Latency function should cover all the mechanisms ensuring the Deterministic Latency such as the existing queuing and scheduling mechanisms and other mechanisms which may be proposed in the future.

It indicates that a type of mechanisms is used for DetNet when one flag is set to 1 and multiple cooperating queuing mechanisms may be implemented when more than one flag is set to 1. For example, Credit-Based Shaper with Asynchronous Traffic Shaping can be used to provide the guaranteed delay. This document proposed the flags to cover the existing mechanisms and other mechanisms should be taken into considerations in the future.

The Flags field is designed as follow:

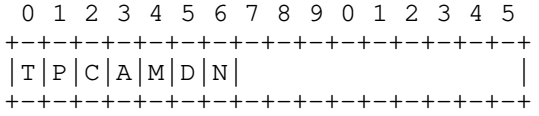


Figure 2: Flag

T flag: 1 bit, if the flag is set to 1, it indicates the TAS (Time Aware Shaping) [IIEEE802.1Qbv] queuing mechanism.

P Flag: 1 bit, if the flag is set to 1, it indicates the CBS (Credit-Based Shaper) [IEEE802.1Q-2014] queuing mechanism.

C Flag: 1 bit, if the flag is set to 1, it indicates the CQF (Cyclic Queuing and Forwarding) [IEEE802.1Qch] queuing mechanism.

A Flag: 1 bit, if the flag is set to 1, it indicates the ATS (Asynchronous Traffic Shaping) [IEEE802.1Qcr] queuing mechanism.

M Flag: 1 bit, if the flag is set to 1, it indicates the Multiple Cyclic Queuing as defined in [I-D.dang-queuing-with-multiple-cyclic-buffers].

D Flag: 1 bit, if the flag is set to 1, it indicates the Deadline-based Queuing mechanisms defined in [I-D.peng-detnet-deadline-based-forwarding] and [I-D.stein-srtsn].

N Flag: 1 bit, if the flag is set to 1, it indicates the ADN mechanism defined in [I-D.joung-detnet-asynch-detnet-framework].

3.1.2. Deterministic Latency Parameters

The Deterministic Latency Parameters sub-TLVs is optional and it provides Deterministic Latency additional parameters for a node to forward a DetNet flow. The format of Sub-TLVs is based on the type and the flag field.

When the M flag is set to 1, the Cycle Sub-TLV may be carried and designed as follow:

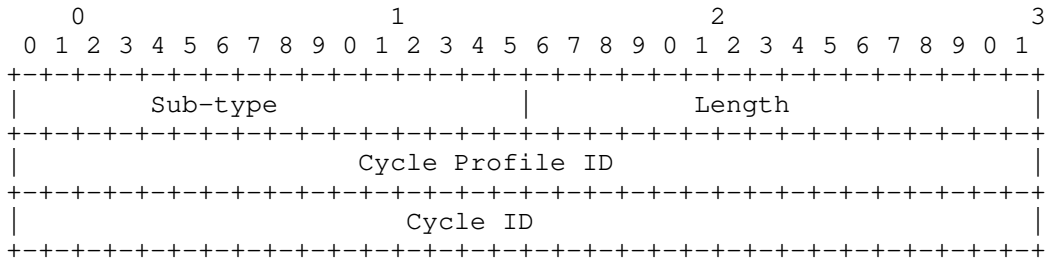


Figure 3: Cycle Sub-TLV

Sub-type (16bits): TBD2, indicates the type of Cycle Sub-TLV.

Length (16bits): indicated the length of Cycle Sub-TLV.

Cycle Profile ID (32bits): indicates the profile ID which the cyclic queue applied at a node.

Cycle ID (32bits): indicates the Cycle ID for a node to forward a DetNet flow.

When the D flag is set to 1, the Deadline Sub-TLV may be carried and designed as follow:

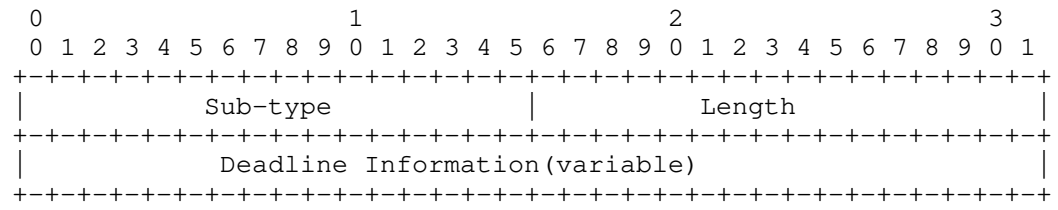


Figure 4: Deadline Sub-TLV

Sub-type (16bits): TBD3, indicates the type of Deadline Sub-TLV.

Length (16bits): indicated the length of Deadline Sub-TLV.

Deadline Information(variable): indicates the deadline information such as deadline as defined in [I-D.stein-srtsn] or planned Deadline and deadline Deviation as defined in [I-D.peng-6man-deadline-option].

4. Encapsulation of DetNet Options

4.1. IPv6 Networks

The DetNet Deterministic Latency information Option is intended to be placed in an IPv6 HbH EH since it must be processed by every DetNet forwarding node along the path. All DetNet forwarding nodes can use the queuing information to achieve the packet forwarding and queue scheduling.

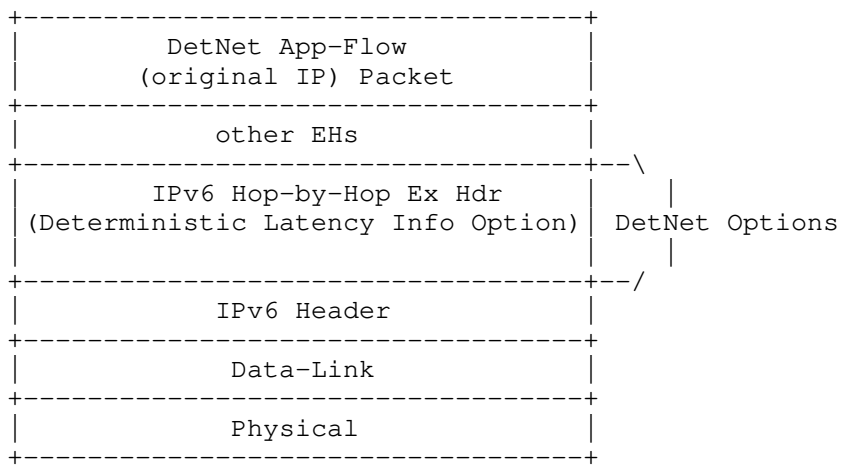


Figure 5: Deterministic Latency Information Option Format

4.2. SRv6 Networks

The DetNet Deterministic Latency information Option is intended to be placed in an DOH EH before an SRH since it must be processed by the DetNet forwarding nodes of the SRv6 segment list. The DetNet forwarding nodes among SRv6 segment list can use the Deterministic Latency to achieve the packet forwarding and queue scheduling.

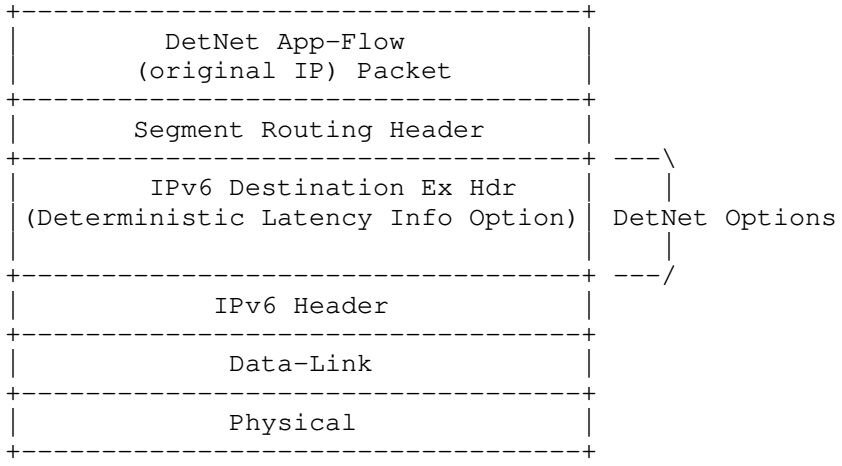


Figure 6: Deterministic Latency Information Option Format

5. Security Considerations

TBA

6. Acknowledgements

The authors would like to thank Peng Liu, Bin Tan, Shaofu Peng for their review, suggestions and comments to this document.

7. IANA Considerations

7.1. New Option for IPv6

This specification updates the "Destination Options and Hop-by-Hop Options" under the "Internet Protocol Version 6 (IPv6) Parameters" registry with the values below:

Type	Description	Reference
TBD1	Deterministic Latency Information Option	[this document]

Table 1

The Sub-TLVs has been proposed for Deterministic Latency Information Option as following shown:

Sub-type	Description	Reference
TBD2	Cycle Sub-TLV	[this document]
TBD3	Deadline Sub-TLV	[this document]

Table 2

8. Normative 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.ietf-detnet-bounded-latency]
Finn, N., Boudec, J. L., 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.joung-detnet-asynch-detnet-framework]
Joung, J., Ryoo, J., Cheung, T., Li, Y., and P. Liu, "Asynchronous Deterministic Networking Framework for Large-Scale Networks", Work in Progress, Internet-Draft, draft-joung-detnet-asynch-detnet-framework-00, 26 June 2022, <https://www.ietf.org/archive/id/draft-joung-detnet-asynch-detnet-framework-00.txt>.

[I-D.liu-detnet-large-scale-requirements]
Liu, P., Li, Y., Eckert, T., Xiong, Q., and J. Ryoo, "Requirements for Large-Scale Deterministic Networks", Work in Progress, Internet-Draft, draft-liu-detnet-large-scale-requirements-02, 10 April 2022, <https://www.ietf.org/archive/id/draft-liu-detnet-large-scale-requirements-02.txt>.

[I-D.peng-6man-deadline-option]
Peng, S. and B. Tan, "Deadline Option", Work in Progress, Internet-Draft, draft-peng-6man-deadline-option-00, 11 January 2022, <https://www.ietf.org/archive/id/draft-peng-6man-deadline-option-00.txt>.

[I-D.peng-detnet-deadline-based-forwarding]
Peng, S., Tan, B., and P. Liu, "Deadline Based Deterministic Forwarding", Work in Progress, Internet-Draft, draft-peng-detnet-deadline-based-forwarding-01, 1 March 2022, <https://www.ietf.org/archive/id/draft-peng-detnet-deadline-based-forwarding-01.txt>.

[I-D.stein-srtsn]
Stein, Y. (., "Segment Routed Time Sensitive Networking", Work in Progress, Internet-Draft, draft-stein-srtsn-01, 29 August 2021, <https://www.ietf.org/archive/id/draft-stein-srtsn-01.txt>.

[I-D.xiong-detnet-large-scale-enhancements]
Xiong, Q. and Z. Du, "DetNet Enhancements for Large-Scale Deterministic Networks", Work in Progress, Internet-Draft, draft-xiong-detnet-large-scale-enhancements-00, 24 February 2022, <https://www.ietf.org/archive/id/draft-xiong-detnet-large-scale-enhancements-00.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2212] Shenker, S., Partridge, C., and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, DOI 10.17487/RFC2212, September 1997, <https://www.rfc-editor.org/info/rfc2212>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[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>.

[RFC8938] Varga, B., "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan
Hubei, 430223
China
Email: xiong.quan@zte.com.cn

Aihua Liu
ZTE Corporation
Shenzhen
China
Email: liu.aihua@zte.com.cn

DetNet
Internet-Draft
Intended status: Standards Track
Expires: 22 December 2022

Y. Li
S. Ren
G. Li
F. Yang
Huawei Technologies
J. Ryoo
ETRI
P. Liu
China Mobile

24 October 2022

IPv6 Options for Cyclic Queuing and Forwarding Variants
draft-yizhou-detnet-ipv6-options-for-cqf-variant-01

Abstract

The fundamental Cyclic Queuing and Forwarding (CQF) defined by Time-Sensitive Networking (TSN) requires no per-stream per-hop state maintenance and at the same time its end-to-end bounded latency and jitter can be easily computed. Such features are attractive and therefore CQF is being considered in wider deployments. To accommodate the different deployments, there are variants of CQF enhancement. This document introduces a new IPv6 option to include the cycle identification to help leverage CQF variants in DetNet network to facilitate the deployments.

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 22 December 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

The latency guarantee is the one of the most important features to be provided by Deterministic Networking (DetNet). [I-D.ietf-detnet-bounded-latency] presents some examples of queuing mechanisms to show how each or the combination of them can be used to achieve the different levels of end-to-end latency bound requirements. Among them, Cyclic Queuing and Forwarding (CQF) shows a great simplicity in calculation and configuration of latency bounds.

Based on the two-buffer CQF specified in annex T of [IEEE8021Q], the latency bound is between (h-1) * T_c + DT and (h+1) * T_c where h is the number of hops, T_c is the cycle interval of buffer swapping and DT, called dead time is a time interval at the end of a cycle, during which no frames can be transmitted from the buffer to ensure the last byte of the cycle to be received earlier than the end of the same

cycle by the next downstream node [I-D.ietf-detnet-bounded-latency]. DT is usually the sum of output delay, link delay, frame preemption delay, and processing delay. There can also be other factors contributing to DT such as the time synchronization drifting. Normally the DT is much lesser than cycle interval T_c so that it is negligible. Hence the minimum latency becomes (h-1) * T_c approximately. That is to say the end-to-end jitter is bounded by 2*T_c approximately for two-buffer CQF.

The latency and jitter bound of CQF is relatively intuitive and almost fully depends on cycle interval T_c and number of hops. The lesser T_c is, the smaller the end-to-end latency is. At the same time, it requires no per-stream per-hop status at the intermediate nodes as long as the cycle interval T_c is carefully selected to make sure the overall bandwidth allowed in T_c is large enough to accommodate all the traffic volume requiring CQF scheduling. Therefore, CQF offers very attractive features to users in terms of simplicity and intuition and is getting wider acceptance in DetNet.

When employing CQF in networks to support higher speed long links, there are variants to enhance CQF for those specific network characteristics, including more buffers (greater than two buffers) and cycle identification. The CQF variants do not change the basic concept of fundamental CQF's en-queue and de-queue logic, while a new data plane option is required. This document introduces the new IPv6 [RFC8200] options to help such CQF variants unambiguously identify the cycle in which the upstream node sends the data packet when the large number of buffers is employed so that CQF variants can adapt to the wider deterministic networking requirements.

2. Terminologies

CQF: Cyclic Queuing and Forwarding. A queuing mechanism defined by annex T of [IEEE8021Q]. This document also uses CQF to refer to the variants enhanced from it using time cycle based en-queuing and draining.

3. Features of CQF Variants

The fundamental CQF uses two buffers to swap the packet receiving and transmission. Swapping is executed every cycle interval T_c. Two buffer generally maps to two traffic class queues.

When CQF is used in higher speed and/or longer links in wider area networking, new features like larger number (>=3) of buffers and cycle identification are introduced. Section 3 and Section 4 illustrate why these features are required and how they work.

3.1. Desire to support higher speed links in DetNet

The fundamental CQF typically uses no less than hundreds of microseconds as a cycle interval. In a network with a small diameter, say less than 8 hops, it is sufficiently good to provide an end-to-end latency bound in the order of several milliseconds.

With the increasing of link speed from 100Mbps to 1Gbps, 10Gbps or even higher in larger networks, either more bytes can be transmitted within the same cycle interval or the smaller cycle interval is required to transmit the same amount of bytes in a cycle as that in low speed networks.

Figure 1 shows a simple calculation on the number of bytes that can be transmitted in a cycle with different cycle intervals and link speeds. 1500 bytes is labeled with * as a baseline because a typical maximum Ethernet frame is 1500 bytes and a selected cycle interval should at least allow one such frame size to be transmitted unless otherwise specified.

Cycle Time	Bytes Transmitted in a Cycle			
	Link Speed			
(us)	100Mbps	1Gbps	10Gbps	
1	12.5	125	1250	
1.2	15	150	1500*	
2	25	250	2500	
4	50	500	5000	
10	125	1250	12500	
12	150	1500*	15000	
120	1500*	15000	150000	

Figure 1: Bytes transmitted within one cycle interval

When the link speed is at 10Gbps, the cycle interval could be as small as 1.2 us if a 1500 byte frame needs to be transmitted in one cycle interval. It is not an accurate calculation because there are certainly other factors to determine the cycle interval. However, it shows that as the link speed increases, cycle interval can be greatly

reduced in practice while satisfying the minimum amount of data transmitted in a single cycle. The end-to-end latency bound when applying CQF is determined by cycle interval and number of hops. That is to say, CQFs with a smaller cycle interval have the potential to meet more strict end-to-end latency requirements in higher link speed networks or meet the same end-to-end latency requirement in networks with much larger network diameter (number of hops).

Industry automation has some typical application period requirement, e.g. 100 us to 2 ms for isochronous traffic, 500 us to 1 ms for cyclic-synchronous and 2 to 20 ms for cyclic-asynchronous traffic. The network cycle interval is usually a fraction of the application period. When the cycle interval is in the order of tens of microseconds, CQF can be used to meet the most strict end-to-end latency requirements. For instance, if we assume the number of hops is 24, when cycle interval is set to 10us, the end-to-end latency bound can be around (24+1)*10 = 250 us which has the potential to meet the latency bound requirement for isochronous traffic.

In summary a higher speed network makes the shorter cycle interval feasible because sufficiently large traffic volume can be transmitted within one cycle interval. A shorter cycle interval further offers shorter end-to-end latency and jitter bounds which provide CQF with the potentials to meet more strict latency requirements in wider deployments while preserving its simplicity of latency calculation and provisioning. Therefore there is a strong motivation to leverage CQF and at the same time to make cycle interval as short as possible.

3.2. Desire to support longer links in DetNet

As shown in Figure 2 for fundamental two buffer CQF, the last byte sent by node A in cycle (i-1) has to be ready for sending at node B before the start of cycle i. To realize it, DT or dead time is imposed. It is a time interval usually at the end of a cycle so that a node should not send the scheduled CQF packets.

Dead time is at least the sum of the maximum propagation delay to the next node, the maximum processing delay at the next node and the maximum other time variations. Therefore either the longer propagation or longer processing delay makes dead time larger. Packets from DetNet service is likely to be propagated over long links in the wider area. It takes around 5us per kilometer to propagate, i.e. 0.5ms every hundred kilometers. Hence the dead time can be as large as milliseconds or tens of milliseconds in case of hundred kilometers of longer links and larger processing delays. That would make the dead time eat up most of the cycle interval when cycle interval is short (e.g., at the same order or one order higher of magnitude in time as dead time). Then the useful time in a cycle

will be much reduced. In some extreme cases, when the link is long and the cycle interval is set to extremely short, the first packet sent in a cycle by a node will not be possibly received in the same cycle interval at the next node. That makes the useful time in a cycle reaches zero in two buffer CQF. Then two buffer CQF will be no longer suitable.

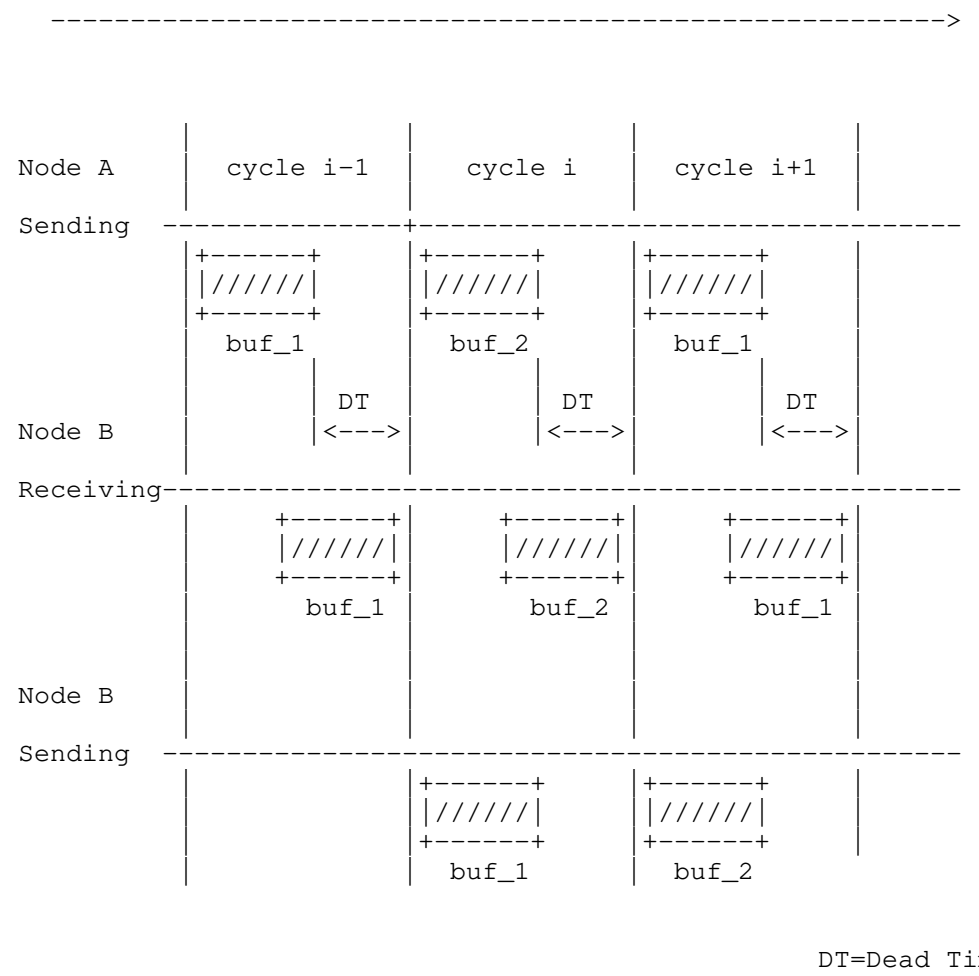


Figure 2: Fundamental Two Buffer CQF

In summary it is hard to achieve the followings at the same time in the fundamental two buffer CQF.

1. Shorter cycle interval to support lower end to end latency as shown in Section 3.1.

2. Larger dead time to support longer link between neighbours.

3. Smaller ratio of dead time to cycle interval to improve utilization.

4. CQF Variants

CQF variants are used to solve the dilemma aforementioned with minor changes to the fundamental two buffer CQF. This section introduces the large number of buffers and cycle identification variants to CQF.

CQF can use more than two buffers to minimize the dead time and increase the useful time in a cycle so as to support long link delay. Figure 3 shows how a three buffer CQF works in a rotating manner in general. Node A sends packets in cycle (i-1). The time interval over which node B receives these packet spans two cycles, cycle (i-1) and cycle i. Hence a method is needed to make node B send them all at once in cycle (i+1) in order to ensure packets in a single cycle from the previous node always being sent out in one cycle at the current node.

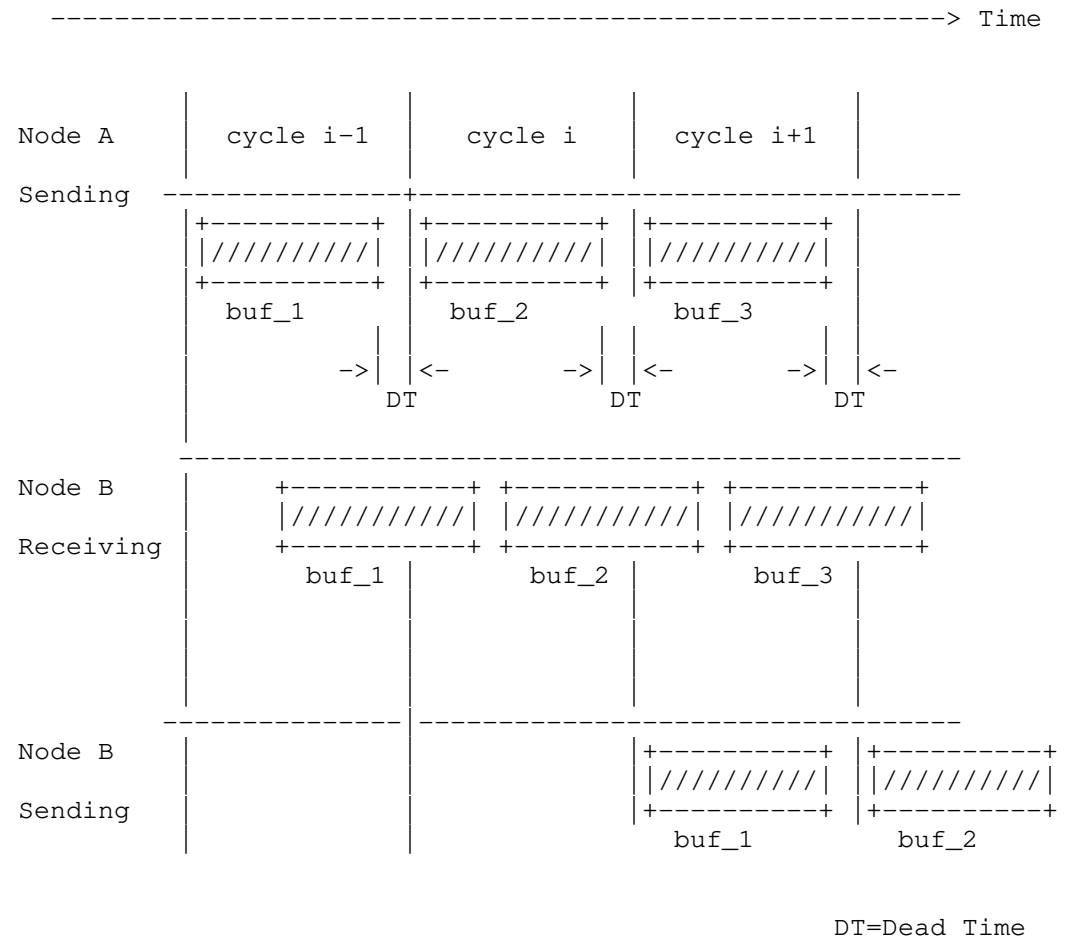


Figure 3: Three Buffer CQF

More than three buffers will be required when the receiving interval at node B for packets sent in a single cycle interval from node A spans over more than two cycle interval boundaries. This can happen when the time variance including propagation, processing, regulation and other factors between two neighbouring DetNet nodes is large. Note that due to the variance in time, the receiving interval at the downstream node can be much larger than one cycle interval in which the upstream node transmits. When time variance is large and cycle interval and dead time are set small, the possible receiving time of the last few packets from node A's cycle (i-1) at node B can overlap with the possible receiving time of the first few packets from node A's cycle i in different rounds of buffer rotations.

Hence, when the buffer number is larger than two, if the receiving side still uses the traditional CQF implicit time borderline to demarcate the receiving packets from the consecutive cycles of the upstream node, it may cause the ambiguity in identifying the right sending cycle at the upstream node and further affect the correctness of the decision of which output buffer to put the received packets at the current node.

Figure 4 shows such an ambiguity when time based cycle demarcation is used. The packet sent by node A in its cycle (i-1) can be received at any time in the receiving interval indicated as "receiving window for A's buf_1" in Figure 4. The receiving window refers to the time interval between the earliest time that the first packet sent in a given cycle from an upstream node is processed and enqueued in an output buffer and the latest time that the last packet of the cycle is processed and enqueued in an output buffer. Network operators may configure the size of the receiving window, taking the time variance of their networks into account. It can be seen that the spanning time period of receiving window is longer than the cycle interval. This is because there is a large time variance experienced between A and B, e.g. varying processing time for different packets in different cycles. It does not mean the receiving interval for every cycle always constantly span over such a large receiving window. The receiving window time interval indeed is determined by the worst case time variance value and that should be used for regular time cycle demarcation.

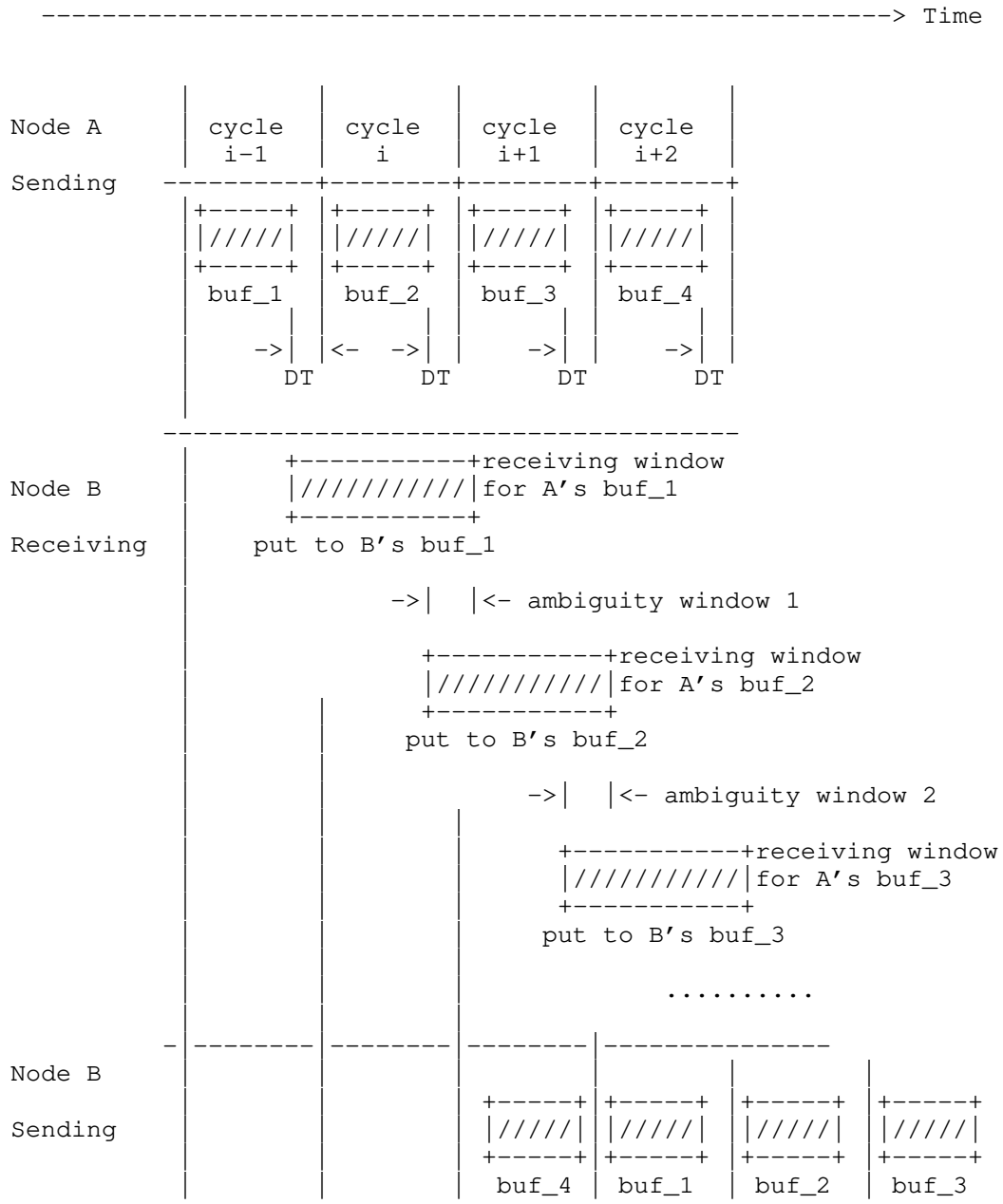


Figure 4: Ambiguity of time based cycle demarcation in CQF

When a packet is received in ambiguity window 1 in Figure 4, node B is not able to use the receiving time to determine which buffer is the correct one to put the packet in because it cannot tell if the packet is sent from cycle (i-1) or cycle i on node A. If node B puts the packet to the wrong output buffer, the packet may experience the unexpected delay. At the same time, the packet occupying the non-designated buffer may break the contracts between the end hosts and DetNet networks and then cause the unpredictable consequences.

It has been noted that the DT can be greatly increased to beat the time variance in order to make the receiving windows do not overlap so as to remove such ambiguity. However, it is not always practical and usually not desired because large DT will eat useful cycle time and bring the low utilization issue as illustrated in Section 3. Therefore, it would be desired to keep DT as small as possible and at the same time identify the cycle interval correctly.

CQF variant carries the cycle identification in the data plane to help determine the correct output port buffer to place the data packets. It requires the IPv6 data plane enhancement which is introduced in next section. The configuration and forwarding logic makes no change from the fundamental CQF. The mapping relation from the upstream node A's output port cycle to the downstream node B's output port cycle should be determined in advance and stored on node B. Such CQF variants can be deployed in networks supporting frequency synchronization only, which alleviate the dependency on network-wide strict time synchronization. When calculating and determining the mapping relation, the time phase difference between neighboring nodes should be taken into consideration if only frequency synchronization is in use. Optionally the mapping relation can be dynamically changed when the network condition changes.

This document does not specify the mechanisms to determine the mapping relation since it can be easily deduced from fundamental CQF and does not require additional standardization. Some example can be found in section 5.1 in [multipleCQF].

5. The CQF-Variant Option

This document defines a new IPv6 option for DetNet to leverage the CQF variant queuing mechanism. This option is to be placed in the IPv6 HbH (Hop-by-Hop) Options or DOH (Destination Option Header) header.

5.1. CQF-Variant Option Format

The CQF-Variant Option helps the receiving port to identify in which time cycle interval the packet is sent from the upstream node. It can be used to determine the output port buffer to enqueue the packet.

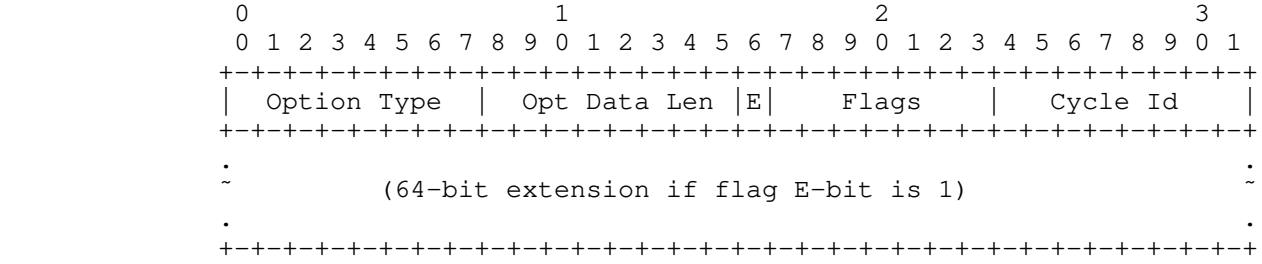


Figure 5: CQF-Variant Option Format Example

CQF-Variant Option fields:

* Option Type: 8-bit identifier of the type of option. Value TBD by IANA. If the processing IPv6 node does not recognize the Option Type it must discard the packet and return an ICMP message (the highest-order 2 bits = 10). The Option Data of this option may change en route to the packet's final destination (the third-highest-order bit=1).

* Opt Data Len: 8-bit length of the option data.

* Flags: 8-bit field to indicate what CQF-Variant information followed. The leftmost bit is called E-bit. When E-bit set to 1, there is a 64-bit extension in length after Cycle Id.

* Cycle Id: 8-bit field to indicate the time cycle ID at output port of the upstream node when the packet is sent out.

* 64-bit extension: This field contains values required for a particular CQF variant, such as timestamp. This field exists only when E-bit in Flags field is set to one. [Editor's Note: Text will be modified or added as specific uses for this field are identified]

5.2. CQF-Variant Option Processing

A packet carrying CQF-Variant Option with Cycle Id does not change the fundamental cyclic queuing and forwarding behaviors of CQF. At the data plane, the packet transmitted from an output port carries an unambiguous cycle Id. There are different ways to manage the cycle id and output buffer. The simplest one is to map a buffer to a cycle id in a rotating manner. When the buffers rotate for transmission, cycle id also rotates. Packets in one buffer are sent all at once within a cycle interval and carry the same cycle id.

When a router receives a data packet with Cycle Id, it uses the cycle id to enqueue the packet to the correct output buffer that implicitly maps to a local cycle id for output transmission. Therefore the cycle id changes every hop.

Each node has the pre-computed and maintained mapping relation between the incoming cycle id of each input port and the output buffer number of the outgoing port. Such relation is determined in advance by measuring and/or configure the various delay components of the serviced DetNet flows as indicated in Figure 1 in [I-D.ietf-detnet-bounded-latency]. Once the output buffer to place an incoming packet is determined, the cycle id value carried in later packet transmission is decided implicitly.

Cycle Id can be used as an implicit aggregated flow id and also is a quick way to identify the streams requiring DetNet service which provides an alternative to use 6-tuple to identify an IPv6 flow shown in [RFC8938].

6. Encapsulation of CQF-Variant Option

When used in IPv6 networks, the CQF-Variant option can be placed in an HbH extension header or DOH.

Figure 6 shows the encapsulation of CQF-Variant option in HbH extension header. When every DetNet forwarding node along the path is provisioned to use CQF variants as the queuing mechanism, this option should be placed here. If a router does not support this option, it discards the packet and returns an ICMP message.

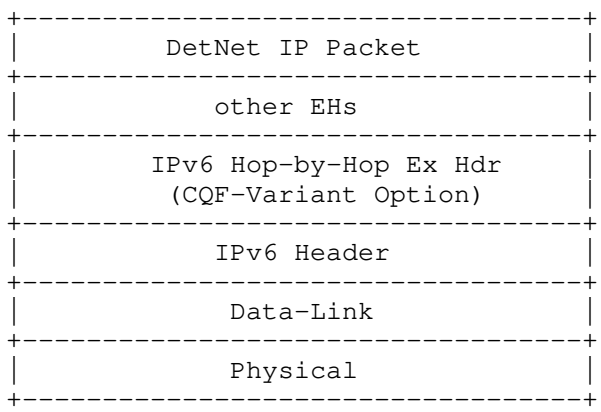


Figure 6: Example 1: CQF-Variant Option Encapsulated in HbH

In some deployments the path selection is indicated using IPv6 routing header (RH) by specifying a set of nodes that must be traversed by the packet along its path to the destination. When such a source routing mechanism is used, CQF-Variant Option is placed in DOH (Destination Option Header) as shown in Figure 7. Then CQF-Variant Option will be processed by the specified in-path routers.

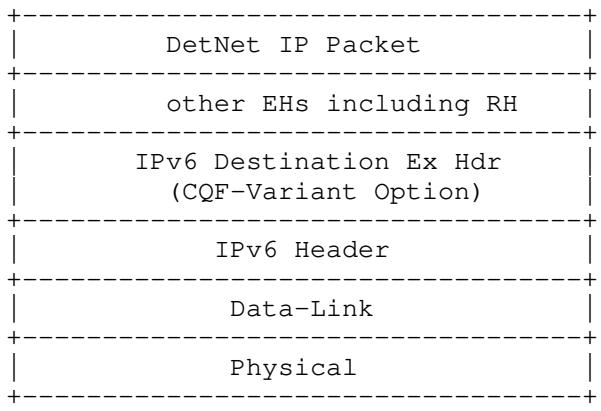


Figure 7: Example 2: CQF-Variant Option Encapsulated in DOH

(To be discussed: Should and how CQF-Variant Option to be placed in SRv6.)

7. Collabration Considerations [Editor's Note: This chapter will be deleted when the WG forms consent on final solution.]

From the comments and presentations in detnet session of IETF 114, the authors paid much attention on related solutions, e.g. [I-D.yzz-detnet-enhanced-data-plane] and [I-D.eckert-detnet-tcqf]. The former document introduced a general IPv6 Option of BLI(Bounded Latency Information) that can be carried in IPv6 extension header. To encapsulate cycle id and potential timestamp information by the way indroduced in BLI, it will takes 5 Bytes and 12 Bytes correspectively see Figure 8. In this document, it will takes 2 Bytes and 10 Bytes for the option data. From the latter TCQF document, it will reuse DSCP code points in a single administrative domain. Only 4 bits of DSCP is available, it is unable to acommodate cycle id or timestamp information. Full discussion should be taken in detnet WG and final decision is significant to future proposal of solution of detnet.

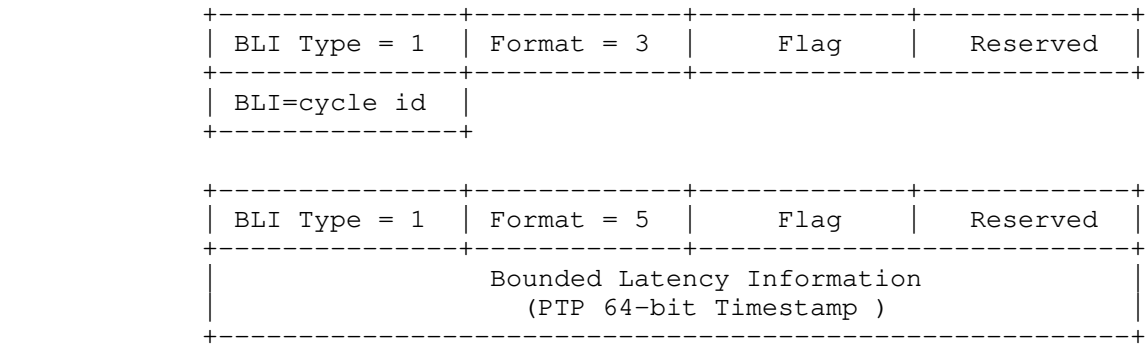


Figure 8: CQF-Variant Information in Format of BLI Options

8. Security Considerations

9. IANA Considerations

This document defines a new CQF-Variant Option for the "Destination Options and Hop-by-Hop Options" under the "Internet Protocol Version 6 (IPv6) Parameters" registry [IPV6-PARMS] with the suggested values in Figure 9.

Hexa	act	chg	rest	Description	Reference
0xB1	10	1	10001	CQF-Variant Option	this document

Figure 9: CQF-Variant Option Code in Destination Options and Hop-by-Hop Options

10. Contributors

The following co-authors have contributed to this document:

Xiaoliang Zheng Huawei Email: zhengxiaoliang@huawei.com

11. References

11.1. Normative References

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

11.2. Informative References

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[I-D.ietf-detnet-bounded-latency] Finn, N., Boudec, J. L., 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.yzz-detnet-enhanced-data-plane] Yang, F., Zhou, T., and L. Zhang, "DetNet Enhanced Data Plane", Work in Progress, Internet-Draft, draft-yzz-detnet-enhanced-data-plane-00, 10 July 2022, <https://www.ietf.org/archive/id/draft-yzz-detnet-enhanced-data-plane-00.txt>.

[I-D.eckert-detnet-tcqf] Eckert, T. T., Bryant, S., and A. G. Malis, "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-00, 5 October 2022, <https://www.ietf.org/archive/id/draft-eckert-detnet-tcqf-00.txt>.

[IEEE8021Q]
"IEEE Standard for Local and Metropolitan Area Network Bridges and Bridged Networks", IEEE Std 802.1Q-2018, DOI 10.1109/ieeestd.2018.8403927, 2018, <https://doi.org/10.1109/ieeestd.2018.8403927>.

[multipleCQF]
Finn, N., "Multiple Cyclic Queuing and Forwarding", October 2021, <https://www.ieee802.org/1/files/public/docs2021/new-finn-multiple-CQF-0921-v02.pdf>.

[IPV6-PARMS]
IANA, "Internet Protocol Version 6 (IPv6) Parameters", n.d., <https://www.iana.org/assignments/ipv6-parameters/ipv6-parameters.xhtml>.

Authors' Addresses

Yizhou Li
Huawei Technologies
Email: liyizhou@huawei.com

Shoushou Ren
Huawei Technologies
Email: renshoushou@huawei.com

Guangpeng Li
Huawei Technologies
Email: liguangpeng@huawei.com

Fan Yang
Huawei Technologies
Email: shirley.yangfan@huawei.com

Jeong-dong Ryoo
ETRI
Email: ryoo@etri.re.kr

Peng Liu
China Mobile
Email: liupengyjy@chinamobile.com

DetNet
Internet-Draft
Intended status: Standards Track
Expires: 24 April 2023

F. Yang
T. Zhou
L. Zhang
Huawei
Z. Du
China Mobile

21 October 2022

DetNet Enhanced Data Plane
draft-yzz-detnet-enhanced-data-plane-01

Abstract

Aiming at providing the bounded latency to DetNet services, DetNet data plane is required to be enhanced. This document provides a method to extend DetNet data plane by introducing the Bounded Latency Information (BLI), which facilitates DetNet transit nodes to guarantee the bounded latency transmission in data plane.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in .

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 24 April 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

DetNet [RFC8655] provides the capability to carry specified unicast or multicast data flows with extremely low data loss rates and bounded end-to-end latency within a network domain. Three primary goals of DetNet QoS are defined in section 3.1 of [RFC8655]:

* Minimum and maximum end-to-end latency from source to destination, timely delivery, and bounded jitter (packet delay variation) derived from these constraints.

* Packet loss ratio under various assumptions as to the operational states of the nodes and links.

* An upper bound on out-of-order packet delivery. It is worth noting that some DetNet applications are unable to tolerate any out-of-order delivery.

To fulfill the goals of DetNet QoS, DetNet architecture [RFC8655] defines a DetNet data plane protocol stack, which includes DetNet forwarding and service sub-layers. Specifically, DetNet data plane framework [RFC8938] specifies two metadata of flow identity and sequence number to be encoded in data plane. Flow-ID is used for identification of the flow or aggregate flow to decide the DetNet traffic treatment and PREOF in both sub-layers. At the same time, sequence number is only used for PREOF in service sub-layer.

For IP DetNet data plane, [RFC8939] specifies a method of using 6-tuple to identify DetNet flows. Management and control information defined in DetNet YANG module [I-D.ietf-detnet-yang] is used to select the forwarding outgoing interface and next hop. It is stated that the allocation of system resources and provisioning of related parameters is used for DetNet traffic treatment. However, [RFC8939] doesn't further specify the related parameters used in data plane.

In [RFC8964], DetNet Control Word (d-CW), DetNet service label (S-Label), and DetNet MPLS forwarding label(s) (F-Label) are defined for the MPLS-based DetNet data plane encapsulation, where the first two information is mainly used for the DetNet service sub-layer functions, the last information is used for the DetNet forwarding sub-layer functions. DetNet controller plane takes the responsibility to provision both flow identification information and the flow-specific resources needed to provide traffic treatment to meet each flow's service requirements. There is no specification in MPLS DetNet data plane to empower the packet treatment capabilities.

There are also other specifications of DetNet data planes such as [RFC9023], [RFC9024], [RFC9025], [RFC9037], and [RFC9056]. These documents specifies the DetNet data planes and interworking technologies of one type of network operating over another sub-network in order to extend the DetNet service range. However, these documents do not introduce new procedure or process, but to follow the specifications defined in [RFC8939] and [RFC8964].

To meet the requirements for large-scale deterministic networks and support the bounded latency objective specified in [I-D.liu-detnet-large-scale-requirements], DetNet data plane is required to be enhanced in the following aspects:

* Explicit inclusion of the metadata used for traffic treatment, especially for bounded latency and jitter, when considering the support of DetNet flows scalability in large scale DetNet networks

* Compatibility to different options of queuing, shaping, policing or any other underlying network technologies, in order to provide bounded latency

* Minimize the end-to-end delay difference of multiple forwarding paths that are used for packet replication and elimination

* DetNet data plane processing of DetNet flow coexists with the non-DetNet flows

This documents provides a method to extend DetNet data plane by introducing Bounded Latency Information (BLI), which facilitates DetNet transit nodes to guarantee the bounded latency transmission in data plane. The resources include the QoS mechanisms, scheduling mechanisms, or any other mechanisms from underlying network layer so as to support bounded latency. This document also proposes a format of bounded latency information and its encapsulations on DetNet data planes.

2. Terminology and Conventions

2.1. Requirement Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Terminology

The abbreviations used in this document are:

BLI: Bounded Latency Information

PREOF: Packet Replication, Elimination, and Ordering Functions

3. Design of DetNet Enhanced Data Plane

In order to support the enhanced traffic treatment functions, such as bounded latency, DetNet data plane is enhanced by carrying a new defined metadata information in DetNet service packets: Bounded Latency Information (BLI).

DetNet uses either one or combination of QoS related and resource allocation technologies to ensure the end-to-end bounded latency. [I-D.ietf-detnet-bounded-latency] introduces a set of scheduling mechanisms can be used to assure the bounded latency. [I-D.stein-srtsn] uses a single stack data structure to provide a unified approach to forwarding and deadline based scheduling. Noted that in most scheduling process, an ancillary information is required to be transmitted between DetNet nodes to facilitate local scheduling. In this document, this ancillary information is named bounded latency information. Bounded latency information is transmitted across multiple DetNet transit nodes and used by the DetNet forwarding sub-layer.

To cope with a variety of scheduling mechanisms and transfer different information in a uniform format in data plane, the bounded latency information is abstracted and classified into two categories: requirement and resource.

3.1. Category 1: Requirement

Bounded latency information in the requirement category may include the information like the end-to-end delay budget, local delay budget, local deadline, delay variation budget, local delay variation budget etc. For example, end-to-end delay budget describes the upper bounded latency value of DetNet flow in network. Then DetNet node may use this information to determine the packet priority or which queue can be used to transmit this packet. Local delay budget is a variation of end-to-end delay budget when multiple DetNet nodes may have same or different delay budget time of each in DetNet network. Deadline is straightforward to indicate how much time is left for this packet to meet the upper bounded latency requirement. Similar practice in 6LoWPANs is given by [RFC9034]. The usage of this information is similar to the delay budget information when DetNet node decides the priority or queue for the packet forwarding. Delay variation [I-D.mohammadpour-detnet-bounded-delay-variation] is another deterministic goal required by DetNet and should be considered in scheduling process when it is required. Priority can also be a type of requirement. DetNet application may assign its priority by different meanings and formats, which may not be equivalently fulfilled by existing QoS priority.

3.2. Category 2: Resource

Bounded latency information in the resource category includes the information like cycle ID, queue ID, and time slot ID etc. Since cycles, queues, or time slots are the real resources can be allocated for DetNet flow, they are named as the time resource ID. For example, time resource ID can represent a cycle ID when cyclic queuing mechanism is used on DetNet node. Time resource ID can also represent a queue ID when queue based scheduling mechanism is locally used on DetNet node. Time resource ID can represent a time slot ID too, when a time slot based mechanism like [RFC9030] is used.

4. Data Field of Bounded Latency Information

This section introduces the data field of bounded latency information in DetNet data plane. The format of the data field is shown as follows.

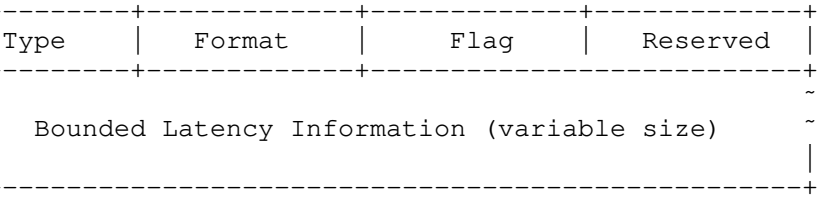


Figure 1: Data Field of Bounded Latency Information

where:

* Bounded Latency Information Type: 8-bit identifier to represent the type of bounded latency information. A new registry is expected to be created and the value is assigned by IANA. Table 1 lists the value of BLI Type and the corresponding Bounded Latency Information defined so far,

BLI Type Value	Bounded Latency Information
0	Reserved
1	Time resource ID
2	Priority
3	End-to-end delay budget
4	Local delay budget
5	End-to-end deadline
6	Local deadline
7	End-to-end delay variation budget
8	Local delay variation budget

Table 1: Bounded Latency Information Type and Value

* Format: 8-bit value to indicate the format of bounded latency information. For example, the format could be 16-bit unsigned integer, 32-bit unsigned integer, PTP or NTP timestamp, and other pre-configured formats. Table 2 lists the value of Format and the corresponding Format defined so far,

Format Value	Format
1	32-bit unsigned Integer
2	16-bit unsigned Integer
3	8-bit unsigned Integer
4	PTP 80-bit Timestamp
5	PTP 64-bit Timestamp
6	NTP 64-bit Timestamp
7	NTP 32-bit Timestamp

Table 2: Format

Bounded Latency Information Type and Format are used together to specify the type, length and format of the bounded latency information.

Reserved: Reserved for future usage.

Time resource ID: the identifier to indicate the underlying resources used for bounded latency. The format is 32-bit unsigned integer.

Priority: QoS priority of the DetNet service packet. As six bits of the Differentiated Services Field [RFC2474] are used as a codepoint (DSCP), the format of priority is 8-bit unsigned integer.

End-to-end delay budget: the end-to-end delay requirement of DetNet service packet. The format is 32-bit unsigned Integer.

Local delay budget: the per hop delay requirement of DetNet service packet on this network node. The format is 32-bit unsigned Integer.

End-to-end deadline: the time when the packet must arrive at the final destination or exit the DetNet network. This time is usually the birth time plus the end-to-end delay budget. The format is the timestamp with proper length.

Local deadline: the time when the packet must exit this network node. The format is the timestamp with proper length.

End-to-end delay variation budget: the end-to-end delay variation requirement of DetNet service packet. The format is 16-bit unsigned Integer.

Local delay variation budget: the per hop delay variation requirement of DetNet service packet on this network node. The format is 16-bit unsigned Integer.

* Flags: 8 bits of flags. A new registry "Bounded Latency Flags" is expected to be created. At the writing time, all flags are unused and undefined.

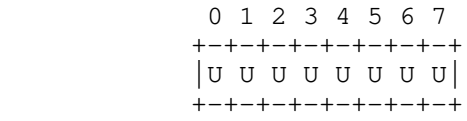


Figure 2: Flag

* Reserved: Keeps zero when it is not specified.

* Bounded Latency Information: indicates the bounded latency information used for local scheduling processing. Table 1 shows the bounded latency information type and the corresponding values. The bounded latency information is different depending on the type of bounded latency information.

5. Encapsulation of Bounded Latency Information

BLI data field can be encapsulated in different DetNet data planes.

5.1. DetNet Data Plane of IP

For IPv6 based DetNet data plane, the data field of bounded latency information is recommended to be carried in IPv6 Extension Header Options, called Bounded Latency Information Option, shown in the following Figure.

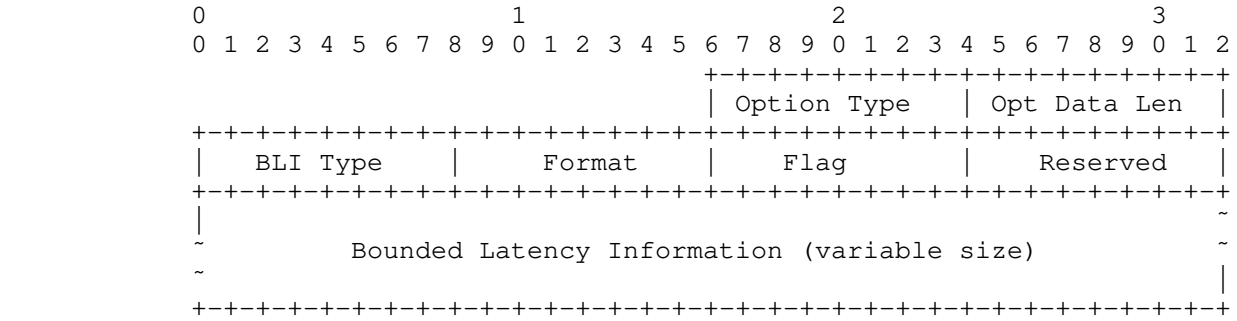


Figure 3: Bounded Latency Information Option

* Option Type: 8-bit identifier of the type of option. Value TBD by IANA; the highest-order 3 bits of this field is 001 to skip over this option and continue processing the header if the processing IPv6 node does not recognize the Option Type and to permit the Option Data may change en route to the destination of packet.

* Opt Data Len: 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

* For Bounded Latency Information data field, see section 4 for details.

Bounded latency information data field is encapsulated in either IPv6 Hop-by-Hop Options header or IPv6 Destination Options header depending on the processing happens at each hop or at the last hop. More than one bounded latency information can appear in one Bounded Latency Information Option. The Option Data Length and the Format are used to locate every bounded latency information. The encapsulation of Bounded Latency Information Option is shown in Figure 4 and Figure 5.

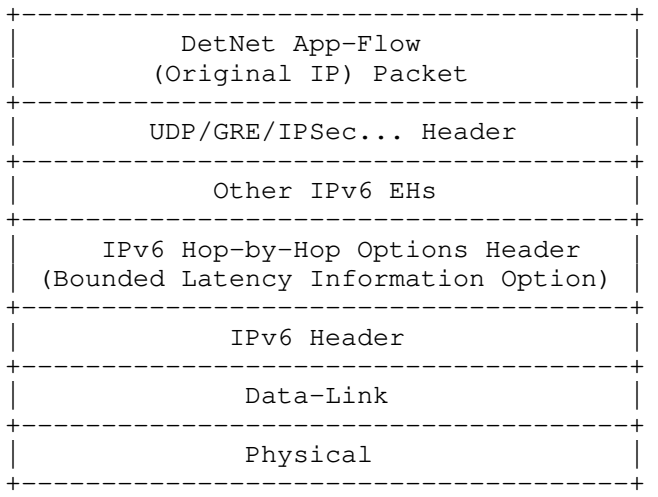


Figure 4: Encapsulation of BLI Option in IPv6 Hop-by-Hop Options Headers

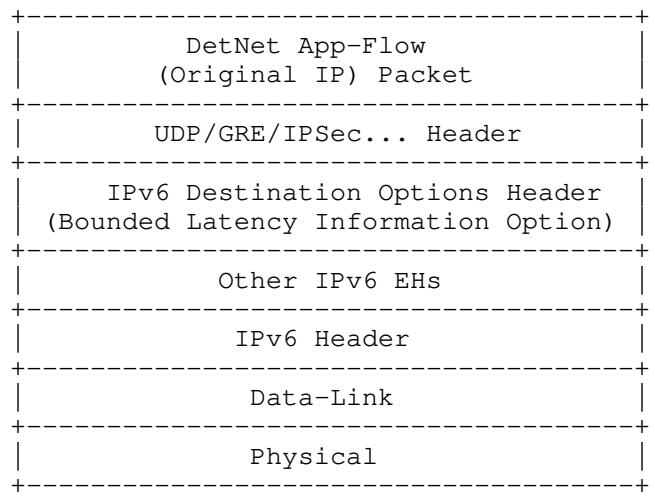


Figure 5: Encapsulation of BLI Option in IPv6 Destination Options Headers

5.2. DetNet Data Plane of MPLS

An MPLS extension header is proposed in [I-D.song-mpls-extension-header]. An MPLS Extension Header (EH) encapsulated with the format of bounded latency information is called Bounded Latency Information Extension Header (BLIEH) and shown in Figure 6.

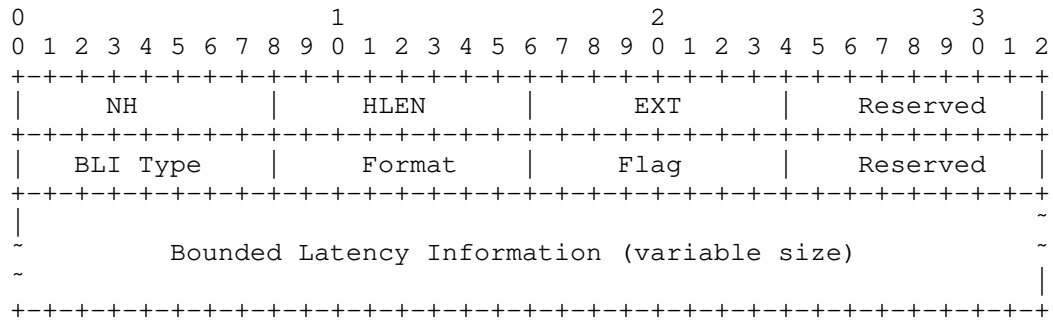


Figure 6: Bounded Latency Information Extension Header

* NH: 8-bit indicator for the Next Header. This field identifies the type of the EH immediately following this EH.

* HLEN: 8-bit unsigned integer for the Extension Header Length in 4-octet units, not including the first 4 octets.

* EXT: 8-bit optional type extension.

The encapsulation of bounded latency information in MPLS extension headers with MPLS label stack is shown in the following figure. More than one BLI can be carried in one Bounded Latency Information Extension Header (BLIEH).

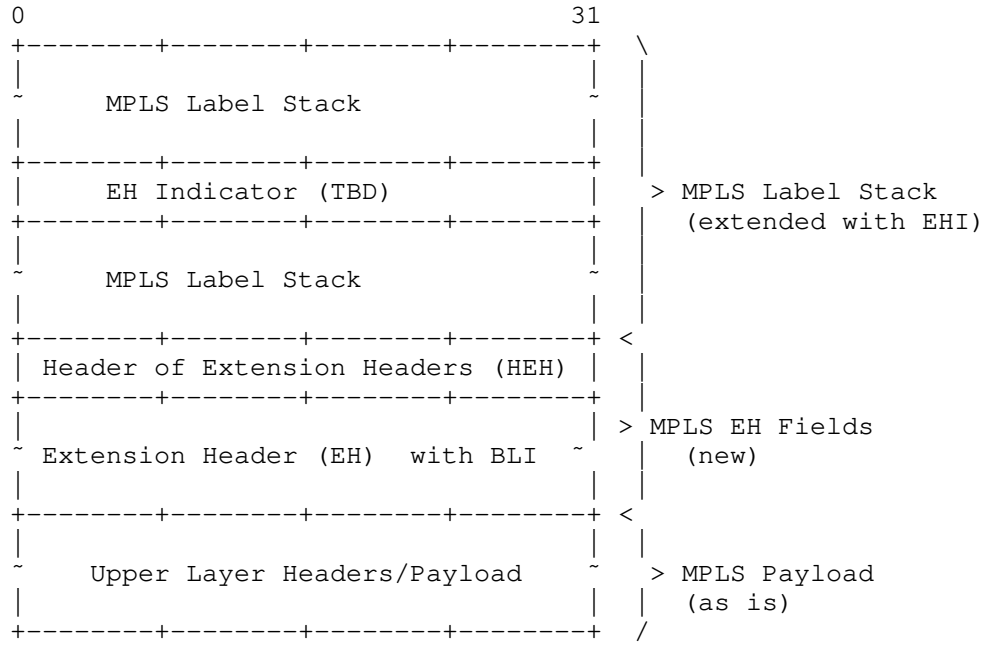


Figure 7: MPLS Encapsulation of Bounded Latency Information Extension Header

5.3. DetNet Data Plane of MPLS over UDP/IP

This document describes a DetNet IP encapsulation that includes the bounded latency information based on the DetNet MPLS over UDP/IP data plane [RFC9025], i.e., leveraging the MPLS-over-UDP technology. The bounded latency guarantee capable DetNet IP encapsulation builds on encapsulating DetNet PW over an IP/UDP tunnel [RFC7510]. It is noted that the format of MPLS Bounded Latency Extension Header (BLIEH) after UDP header is the same with the format of MPLS Bounded Latency Extension Header (BLIEH) defined in section 5.2, as well as without using any MPLS forwarding labels. The encapsulation of bounded

latency information in DetNet Data Plane of MPLS over UDP/IP is shown in the following figure.

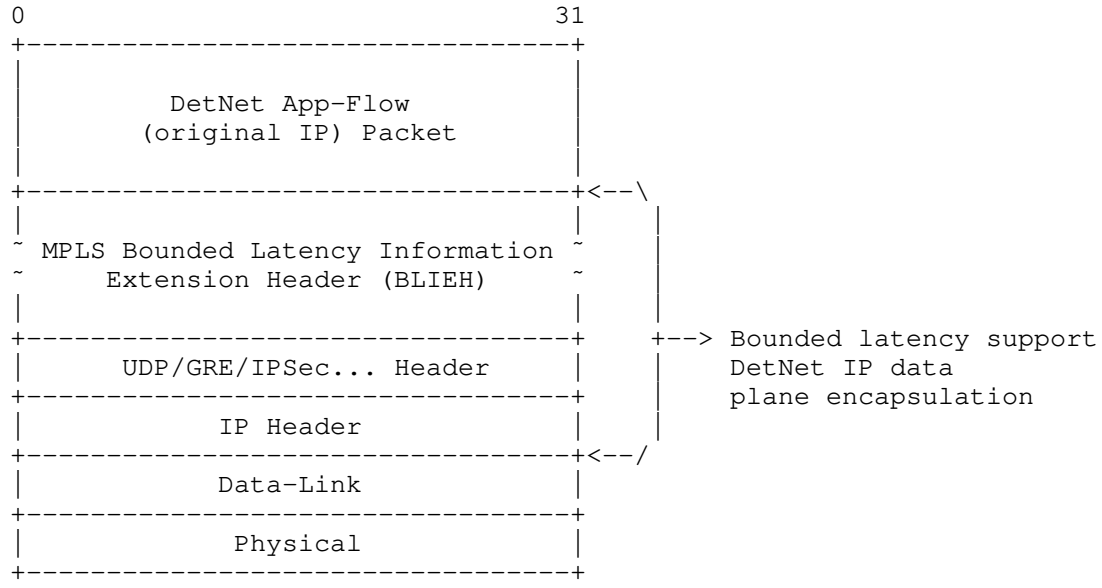


Figure 8: IPv6 extension option of bounded latency

6. IANA Considerations

6.1. New Destination Options and Hop-by-Hop Options

IANA is requested to allocate a value of "Destination Options and Hop-by-Hop Options" under "Internet Protocol Version 6 (IPv6) Parameters" registry. The suggested value is:

Hex	act	chg	rest	Description	Reference
TBD	00	1	TBD	BLI Option	This I-D

Bounded Latency Information Option

6.2. New Type of MPLS Extension Header

IANA is requested to allocate a 8-bit indicator for the Next Header to the Bounded Latency Extension Header.

6.3. New Subregistry of Bounded Latency Information Type

IANA is requested to define a new subregistry of "Bounded Latency Information Type" for the "Bounded Latency Information Option" under "Internet Protocol Version 6 (IPv6) Parameters" registry.

This new subregistry will include the following registries:

Suggested Value	Meaning	Reference
TBD	Reserved	This I-D
TBD	Time resource ID	This I-D
TBD	Priority	This I-D
TBD	End-to-end delay budget	This I-D
TBD	Local delay budget	This I-D
TBD	End-to-end deadline	This I-D
TBD	Local deadline	This I-D
TBD	End-to-end delay variation budget	This I-D
TBD	Local delay variation budget	This I-D

Bounded Latency Information Type

7. Security Considerations

TBD

8. References

8.1. Normative References

[I-D.ietf-detnet-bounded-latency]
Finn, N., Boudec, J. L., 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.ietf-detnet-yang]
Geng, X., Ryoo, Y., Fedyk, D., Rahman, R., and Z. Li, "Deterministic Networking (DetNet) YANG Model", Work in Progress, Internet-Draft, draft-ietf-detnet-yang-17, 4 October 2022, <https://www.ietf.org/archive/id/draft-ietf-detnet-yang-17.txt>.

[I-D.liu-detnet-large-scale-requirements]
Liu, P., Li, Y., Eckert, T., Xiong, Q., Ryoo, J., Zhu, S., and X. Geng, "Requirements for Large-Scale Deterministic Networks", Work in Progress, Internet-Draft, draft-liu-detnet-large-scale-requirements-05, 20 October 2022, <https://datatracker.ietf.org/api/v1/doc/document/draft-liu-detnet-large-scale-requirements/>.

[I-D.mohammadpour-detnet-bounded-delay-variation]
Mohammadpour, E. and J. L. Boudec, "DetNet Bounded Packet-Delay-Variation", Work in Progress, Internet-Draft, draft-mohammadpour-detnet-bounded-delay-variation-00, 10 September 2021, <https://www.ietf.org/archive/id/draft-mohammadpour-detnet-bounded-delay-variation-00.txt>.

[I-D.song-mpls-extension-header]
Song, H., Zhou, T., Andersson, L., Zhang, Z. J., and R. Gandhi, "MPLS Network Actions using Post-Stack Extension Headers", Work in Progress, Internet-Draft, draft-song-mpls-extension-header-11, 15 October 2022, <https://www.ietf.org/archive/id/draft-song-mpls-extension-header-11.txt>.

[I-D.stein-srtsn]
Stein, Y. (., "Segment Routed Time Sensitive Networking", Work in Progress, Internet-Draft, draft-stein-srtsn-01, 29 August 2021, <https://www.ietf.org/archive/id/draft-stein-srtsn-01.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[RFC8939] Varga, B., Ed., Farkas, J., Berger, L., Fedyk, D., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP", RFC 8939, DOI 10.17487/RFC8939, November 2020, <https://www.rfc-editor.org/info/rfc8939>.

[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>.

[RFC9023] Varga, B., Ed., Farkas, J., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP over IEEE 802.1 Time-Sensitive Networking (TSN)", RFC 9023, DOI 10.17487/RFC9023, June 2021, <https://www.rfc-editor.org/info/rfc9023>.

[RFC9024] Varga, B., Ed., Farkas, J., Malis, A., Bryant, S., and D. Fedyk, "Deterministic Networking (DetNet) Data Plane: IEEE 802.1 Time-Sensitive Networking over MPLS", RFC 9024, DOI 10.17487/RFC9024, June 2021, <https://www.rfc-editor.org/info/rfc9024>.

[RFC9025] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: MPLS over UDP/IP", RFC 9025, DOI 10.17487/RFC9025, April 2021, <https://www.rfc-editor.org/info/rfc9025>.

[RFC9030] Thubert, P., Ed., "An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)", RFC 9030, DOI 10.17487/RFC9030, May 2021, <https://www.rfc-editor.org/info/rfc9030>.

[RFC9034] Thomas, L., Anamalamudi, S., Anand, S.V.R., Hegde, M., and C. Perkins, "Packet Delivery Deadline Time in the Routing Header for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 9034, DOI 10.17487/RFC9034, June 2021, <https://www.rfc-editor.org/info/rfc9034>.

[RFC9037] Varga, B., Ed., Farkas, J., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: MPLS over IEEE 802.1 Time-Sensitive Networking (TSN)", RFC 9037, DOI 10.17487/RFC9037, June 2021, <https://www.rfc-editor.org/info/rfc9037>.

[RFC9056] Varga, B., Ed., Berger, L., Fedyk, D., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: IP over MPLS", RFC 9056, DOI 10.17487/RFC9056, October 2021, <https://www.rfc-editor.org/info/rfc9056>.

8.2. Informative References

[I-D.peng-detnet-deadline-based-forwarding]
Peng, S., Tan, B., and P. Liu, "Deadline Based Deterministic Forwarding", Work in Progress, Internet-Draft, draft-peng-detnet-deadline-based-forwarding-03, 22 October 2022, <https://www.ietf.org/archive/id/draft-peng-detnet-deadline-based-forwarding-03.txt>.

[I-D.yizhou-detnet-ipv6-options-for-cqf-variant]
Li, Y., Ren, S., Li, G., Yang, F., Ryoo, J., and P. Liu, "IPv6 Options for Cyclic Queuing and Forwarding Variants", Work in Progress, Internet-Draft, draft-yizhou-detnet-ipv6-options-for-cqf-variant-00, 19 June 2022, <https://www.ietf.org/archive/id/draft-yizhou-detnet-ipv6-options-for-cqf-variant-00.txt>.

Appendix A. BLI Examples

The following examples are provided to give instructions on how Bounded Latency Information is used when network node implements different algorithms to guarantee the bounded latency transmission.

A.1. Cycle Based Algorithms

When network node implements cycle based algorithms for example [I-D.yizhou-detnet-ipv6-options-for-cqf-variant] , cycles are the local resources used to guarantee the bounded latency transmission. Cycle ID is expected to be carried in data plane. Thus, the data field of BLI is suggested as follows:

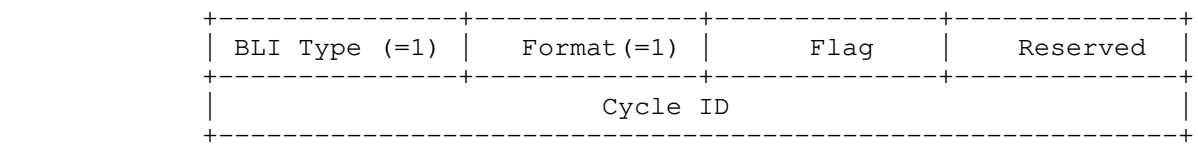


Figure A.1: Data Field of BLI Used With Cycle Based Algorithms

A.2. Time Slot Based Algorithms

When network node implements time slot based algorithms, time slots are the local resources used to guarantee the bounded latency transmission. Time Slot ID is expected to be carried in data plane. Thus, the data field of BLI is suggested as follows:

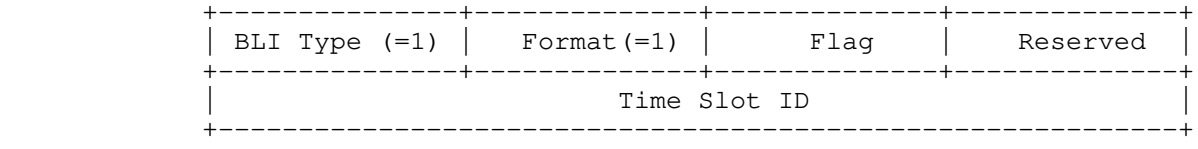


Figure A.2: Data Field of BLI Used With Time Slot Based Algorithms

A.3. Budget Based Algorithms

When network node implements the budget based algorithms to provide bounded latency transmission, end to end or per hop delay budget or delay variation budget information is the requirement proposed from the services and expected to be carried in data plane. The data fields of BLI used with delay budget based algorithms are suggested as follows:

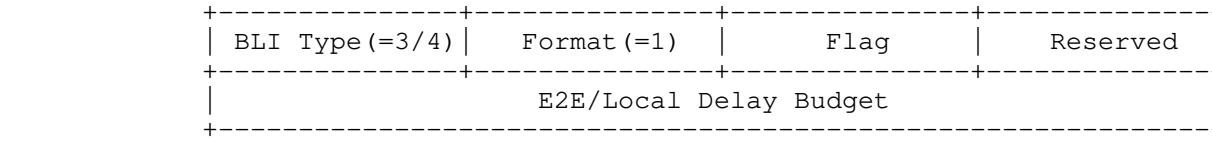


Figure A.3: Data Field of BLI Used With Delay Budget Based Algorithms

The data fields of BLI used with delay variation budget based algorithms are suggested as follows:

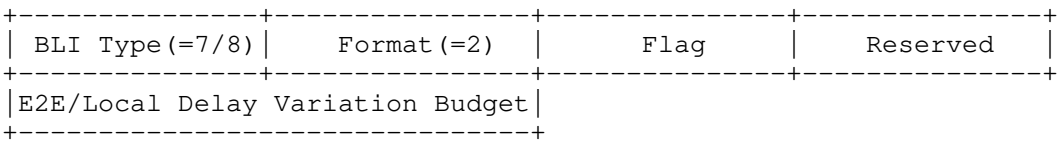


Figure A.4: Data Field of BLI Used With Delay Variation Budget Based Algorithms

A.4. Deadline Based Algorithms

When network node implements deadline based algorithms like EDF, Deadine forwarding [I-D.peng-detnet-deadline-based-forwarding] to provide bounded latency transmission, end to end or per hop packet deadline is the requirement proposed from the services and expected to be carried in data plane. The data fields of BLI used with deadline based algorithms are suggested as follows:

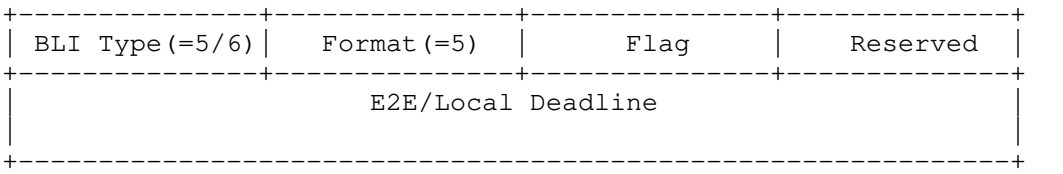


Figure A.5: Data Field of BLI Used With Deadline Based Algorithms

A.5. Priority Based Algorithms

When network node implements priority based algorithms, priority is the requirement proposed from the services. Priority ID is expected to be carried in data plane. The data field of BLI is suggested as follows:

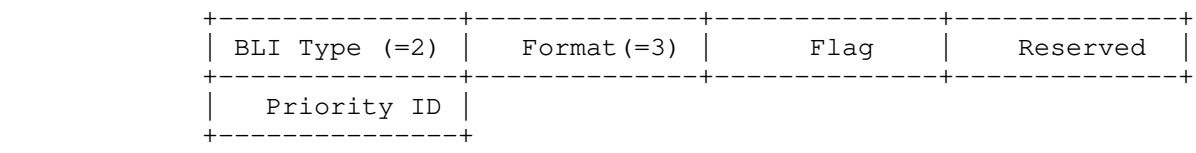


Figure A.6: Data Field of BLI Used With Priority Based Algorithms

Authors' Addresses

Fan Yang
Huawei
156 Beiqing Rd.
Beijing
100095
China
Email: shirley.yangfan@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd.
Beijing
100095
China
Email: zhoutianran@huawei.com

Li Zhang
Huawei
156 Beiqing Rd.
Beijing
100095
China
Email: zhangli344@huawei.com

Zongpeng Du
China Mobile
No.32 XuanWuMen West Street
Beijing
100053
China
Email: duzongpeng@foxmail.com

DetNet
Internet-Draft
Intended status: Standards Track
Expires: 27 June 2023

X. Geng
T. Zhou
L. Zhang
Huawei
Z. Du
China Mobile

24 December 2022

DetNet Enhanced Data Plane
draft-yzz-detnet-enhanced-data-plane-02

Abstract

Aiming at providing the bounded latency to DetNet services, DetNet data plane is required to be enhanced. This document provides a method to extend DetNet data plane by introducing the Bounded Latency Information (BLI), which facilitates DetNet transit nodes to guarantee the bounded latency transmission in data plane.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in .

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 27 June 2023.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

DetNet [RFC8655] provides the capability to carry specified unicast or multicast data flows with extremely low data loss rates and bounded end-to-end latency within a network domain. Three primary goals of DetNet QoS are defined in section 3.1 of [RFC8655]:

* Minimum and maximum end-to-end latency from source to destination, timely delivery, and bounded jitter (packet delay variation) derived from these constraints.

* Packet loss ratio under various assumptions as to the operational states of the nodes and links.

* An upper bound on out-of-order packet delivery. It is worth noting that some DetNet applications are unable to tolerate any out-of-order delivery.

To fulfill the goals of DetNet QoS, DetNet architecture [RFC8655] defines a DetNet data plane protocol stack, which includes DetNet forwarding and service sub-layers. Specifically, DetNet data plane framework [RFC8938] specifies two metadata of flow identity and sequence number to be encoded in data plane. Flow-ID is used for identification of the flow or aggregate flow to decide the DetNet traffic treatment and PREOF in both sub-layers. At the same time, sequence number is only used for PREOF in service sub-layer.

For IP DetNet data plane, [RFC8939] specifies a method of using 6-tuple to identify DetNet flows. Management and control information defined in DetNet YANG module [I-D.ietf-detnet-yang] is used to select the forwarding outgoing interface and next hop. It is stated that the allocation of system resources and provisioning of related parameters is used for DetNet traffic treatment. However, [RFC8939] doesn't further specify the related parameters used in data plane.

In [RFC8964], DetNet Control Word (d-CW), DetNet service label (S-Label), and DetNet MPLS forwarding label(s) (F-Label) are defined for the MPLS-based DetNet data plane encapsulation, where the first two information is mainly used for the DetNet service sub-layer functions, the last information is used for the DetNet forwarding sub-layer functions. DetNet controller plane takes the responsibility to provision both flow identification information and the flow-specific resources needed to provide traffic treatment to meet each flow's service requirements. There is no specification in MPLS DetNet data plane to empower the packet treatment capabilities.

There are also other specifications of DetNet data planes such as [RFC9023], [RFC9024], [RFC9025], [RFC9037], and [RFC9056]. These documents specifies the DetNet data planes and interworking technologies of one type of network operating over another sub-network in order to extend the DetNet service range. However, these documents do not introduce new procedure or process, but to follow the specifications defined in [RFC8939] and [RFC8964].

To meet the requirements for large-scale deterministic networks and support the bounded latency objective specified in [I-D.liu-detnet-large-scale-requirements], DetNet data plane is required to be enhanced in the following aspects:

* Explicit inclusion of the metadata used for traffic treatment, especially for bounded latency and jitter, when considering the support of DetNet flows scalability in large scale DetNet networks

* Compatibility to different options of queuing, shaping, policing or any other underlying network technologies, in order to provide bounded latency

* Minimize the end-to-end delay difference of multiple forwarding paths that are used for packet replication and elimination

* DetNet data plane processing of DetNet flow coexists with the non-DetNet flows

This documents provides a method to extend DetNet data plane by introducing Bounded Latency Information (BLI), which facilitates DetNet transit nodes to guarantee the bounded latency transmission in data plane. The resources include the QoS mechanisms, scheduling mechanisms, or any other mechanisms from underlying network layer so as to support bounded latency. This document also proposes a format of bounded latency information and its encapsulations on DetNet data planes.

2. Terminology and Conventions

2.1. Requirement Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Terminology

The abbreviations used in this document are:

BLI: Bounded Latency Information

PREOF: Packet Replication, Elimination, and Ordering Functions

3. Design of DetNet Enhanced Data Plane

In order to support the enhanced traffic treatment functions, such as bounded latency, DetNet data plane is enhanced by carrying a new defined metadata information in DetNet service packets: Bounded Latency Information (BLI).

DetNet uses either one or combination of QoS related and resource allocation technologies to ensure the end-to-end bounded latency. [I-D.ietf-detnet-bounded-latency] introduces a set of scheduling mechanisms can be used to assure the bounded latency. [I-D.stein-srtsn] uses a single stack data structure to provide a unified approach to forwarding and deadline based scheduling. Noted that in most scheduling process, an ancillary information is required to be transmitted between DetNet nodes to facilitate local scheduling. In this document, this ancillary information is named bounded latency information. Bounded latency information is transmitted across multiple DetNet transit nodes and used by the DetNet forwarding sub-layer.

To cope with a variety of scheduling mechanisms and transfer different information in a uniform format in data plane, the bounded latency information is abstracted and classified into two categories: requirement and resource.

3.1. Category 1: Requirement

Bounded latency information in the requirement category may include the information like the end-to-end delay budget, local delay budget, local deadline, delay variation budget, local delay variation budget etc. For example, end-to-end delay budget describes the upper bounded latency value of DetNet flow in network. Then DetNet node may use this information to determine the packet priority or which queue can be used to transmit this packet. Local delay budget is a variation of end-to-end delay budget when multiple DetNet nodes may have same or different delay budget time of each in DetNet network. Deadline is straightforward to indicate how much time is left for this packet to meet the upper bounded latency requirement. Similar practice in 6LoWPANs is given by [RFC9034]. The usage of this information is similar to the delay budget information when DetNet node decides the priority or queue for the packet forwarding. Delay variation [I-D.mohammadpour-detnet-bounded-delay-variation] is another deterministic goal required by DetNet and should be considered in scheduling process when it is required. Priority can also be a type of requirement. DetNet application may assign its priority by different meanings and formats, which may not be equivalently fulfilled by existing QoS priority.

3.2. Category 2: Resource

Bounded latency information in the resource category includes the information like cycle ID, queue ID, and time slot ID etc. Since cycles, queues, or time slots are the real resources can be allocated for DetNet flow, they are named as the time resource ID. For example, time resource ID can represent a cycle ID when cyclic queuing mechanism is used on DetNet node. Time resource ID can also represent a queue ID when queue based scheduling mechanism is locally used on DetNet node. Time resource ID can represent a time slot ID too, when a time slot based mechanism like [RFC9030] is used.

4. Data Field of Bounded Latency Information

This section introduces the data field of bounded latency information in DetNet data plane. The format of the data field is shown as follows.

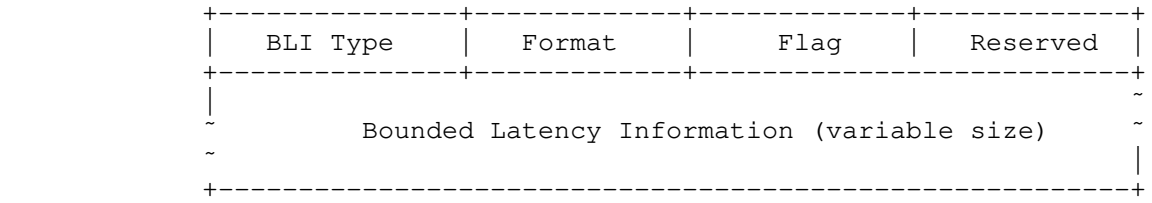


Figure 1: Data Field of Bounded Latency Information

where:

* Bounded Latency Information Type: 8-bit identifier to represent the type of bounded latency information. A new registry is expected to be created and the value is assigned by IANA. Table 1 lists the value of BLI Type and the corresponding Bounded Latency Information defined so far,

BLI Type Value	Bounded Latency Information
0	Reserved
1	Time resource ID
2	Priority
3	End-to-end delay budget
4	Local delay budget
5	End-to-end deadline
6	Local deadline
7	End-to-end delay variation budget
8	Local delay variation budget

Table 1: Bounded Latency Information Type and Value

* Format: 8-bit value to indicate the format of bounded latency information. For example, the format could be 16-bit unsigned integer, 32-bit unsigned integer, PTP or NTP timestamp, and other pre-configured formats. Table 2 lists the value of Format and the corresponding Format defined so far,

Format Value	Format
1	32-bit unsigned Integer
2	16-bit unsigned Integer
3	8-bit unsigned Integer
4	PTP 80-bit Timestamp
5	PTP 64-bit Timestamp
6	NTP 64-bit Timestamp
7	NTP 32-bit Timestamp

Table 2: Format

Bounded Latency Information Type and Format are used together to specify the type, length and format of the bounded latency information.

Reserved: Reserved for future usage.

Time resource ID: the identifier to indicate the underlying resources used for bounded latency. The format is 32-bit unsigned integer.

Priority: QoS priority of the DetNet service packet. As six bits of the Differentiated Services Field [RFC2474] are used as a codepoint (DSCP), the format of priority is 8-bit unsigned integer.

End-to-end delay budget: the end-to-end delay requirement of DetNet service packet. The format is 32-bit unsigned Integer.

Local delay budget: the per hop delay requirement of DetNet service packet on this network node. The format is 32-bit unsigned Integer.

End-to-end deadline: the time when the packet must arrive at the final destination or exit the DetNet network. This time is usually the birth time plus the end-to-end delay budget. The format is the timestamp with proper length.

Local deadline: the time when the packet must exit this network node. The format is the timestamp with proper length.

End-to-end delay variation budget: the end-to-end delay variation requirement of DetNet service packet. The format is 16-bit unsigned Integer.

Local delay variation budget: the per hop delay variation requirement of DetNet service packet on this network node. The format is 16-bit unsigned Integer.

* Flags: 8 bits of flags. A new registry "Bounded Latency Flags" is expected to be created. At the writing time, all flags are unused and undefined.

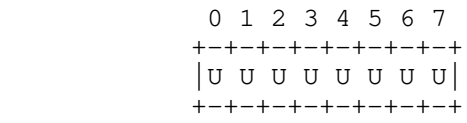


Figure 2: Flag

* Reserved: Keeps zero when it is not specified.

* Bounded Latency Information: indicates the bounded latency information used for local scheduling processing. Table 1 shows the bounded latency information type and the corresponding values. The bounded latency information is different depending on the type of bounded latency information.

5. Encapsulation of Bounded Latency Information

BLI data field can be encapsulated in different DetNet data planes.

5.1. DetNet Data Plane of IP

For IPv6 based DetNet data plane, the data field of bounded latency information is recommended to be carried in IPv6 Extension Header Options, called Bounded Latency Information Option, shown in the following Figure.

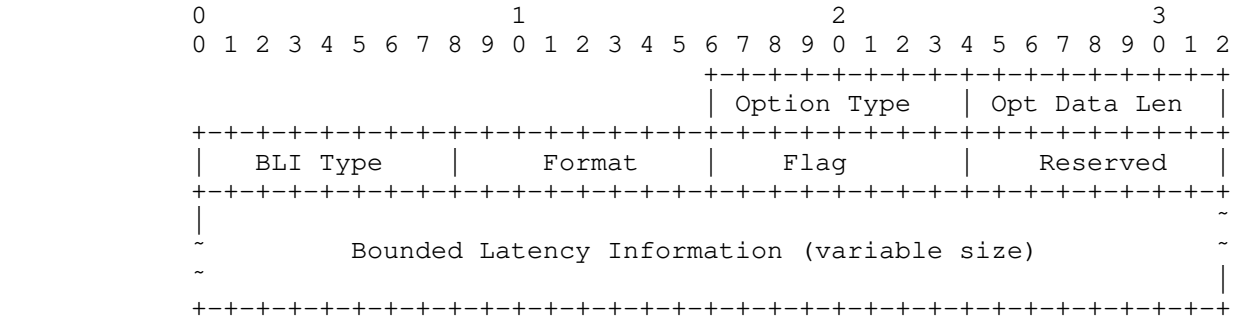


Figure 3: Bounded Latency Information Option

* Option Type: 8-bit identifier of the type of option. Value TBD by IANA; the highest-order 3 bits of this field is 001 to skip over this option and continue processing the header if the processing IPv6 node does not recognize the Option Type and to permit the Option Data may change en route to the destination of packet.

* Opt Data Len: 8-bit unsigned integer. Length of the Option Data field of this option, in octets.

* For Bounded Latency Information data field, see section 4 for details.

Bounded latency information data field is encapsulated in either IPv6 Hop-by-Hop Options header or IPv6 Destination Options header depending on the processing happens at each hop or at the last hop. More than one bounded latency information can appear in one Bounded Latency Information Option. The Option Data Length and the Format are used to locate every bounded latency information. The encapsulation of Bounded Latency Information Option is shown in Figure 4 and Figure 5.

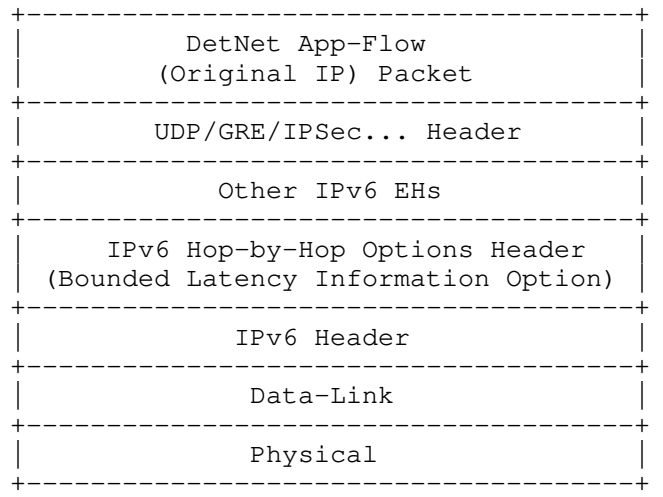


Figure 4: Encapsulation of BLI Option in IPv6 Hop-by-Hop Options Headers

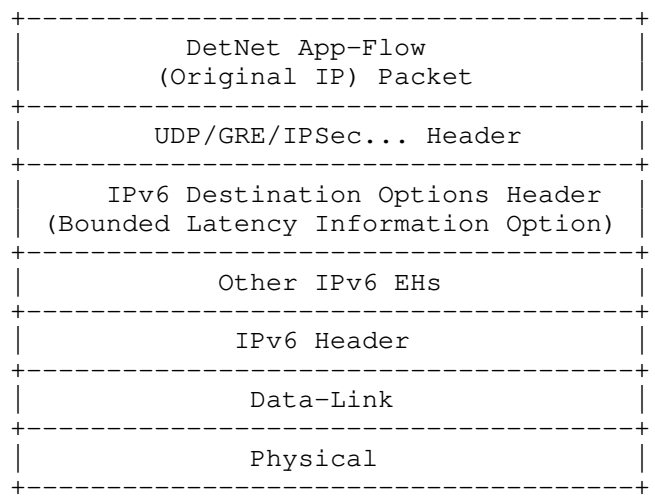


Figure 5: Encapsulation of BLI Option in IPv6 Destination Options Headers

5.2. DetNet Data Plane of MPLS

An MPLS extension header is proposed in [I-D.song-mpls-extension-header]. An MPLS Extension Header (EH) encapsulated with the format of bounded latency information is called Bounded Latency Information Extension Header (BLIEH) and shown in Figure 6.

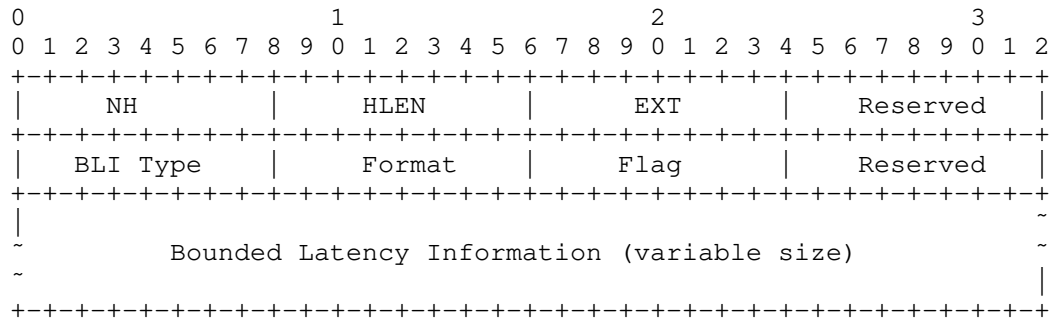


Figure 6: Bounded Latency Information Extension Header

* NH: 8-bit indicator for the Next Header. This field identifies the type of the EH immediately following this EH.

* HLEN: 8-bit unsigned integer for the Extension Header Length in 4-octet units, not including the first 4 octets.

* EXT: 8-bit optional type extension.

The encapsulation of bounded latency information in MPLS extension headers with MPLS label stack is shown in the following figure. More than one BLI can be carried in one Bounded Latency Information Extension Header (BLIEH).

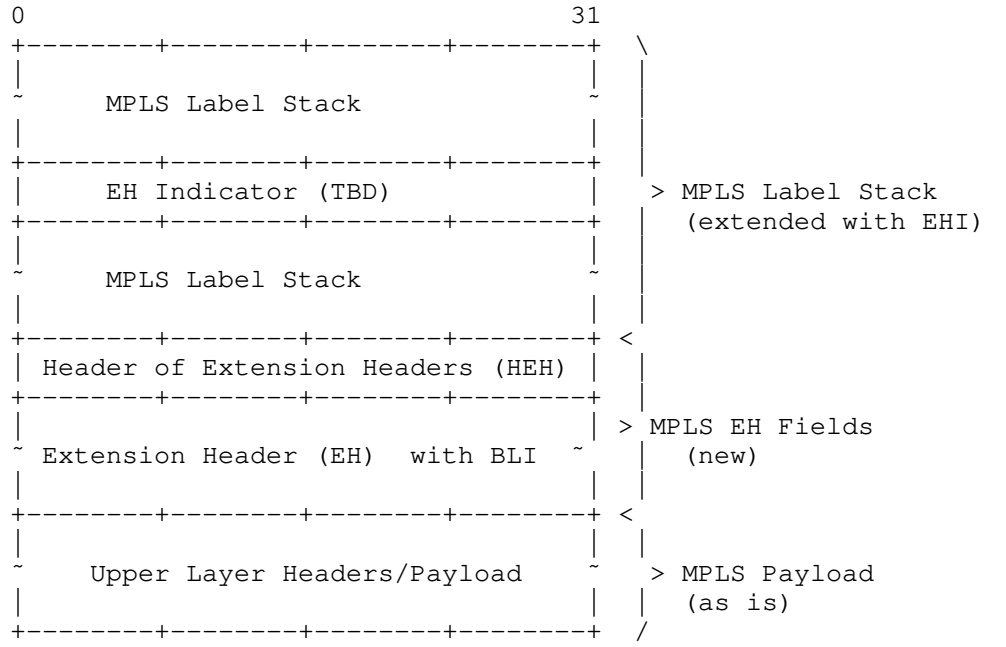


Figure 7: MPLS Encapsulation of Bounded Latency Information Extension Header

5.3. DetNet Data Plane of MPLS over UDP/IP

This document describes a DetNet IP encapsulation that includes the bounded latency information based on the DetNet MPLS over UDP/IP data plane [RFC9025], i.e., leveraging the MPLS-over-UDP technology. The bounded latency guarantee capable DetNet IP encapsulation builds on encapsulating DetNet PW over an IP/UDP tunnel [RFC7510]. It is noted that the format of MPLS Bounded Latency Extension Header (BLIEH) after UDP header is the same with the format of MPLS Bounded Latency Extension Header (BLIEH) defined in section 5.2, as well as without using any MPLS forwarding labels. The encapsulation of bounded

latency information in DetNet Data Plane of MPLS over UDP/IP is shown in the following figure.

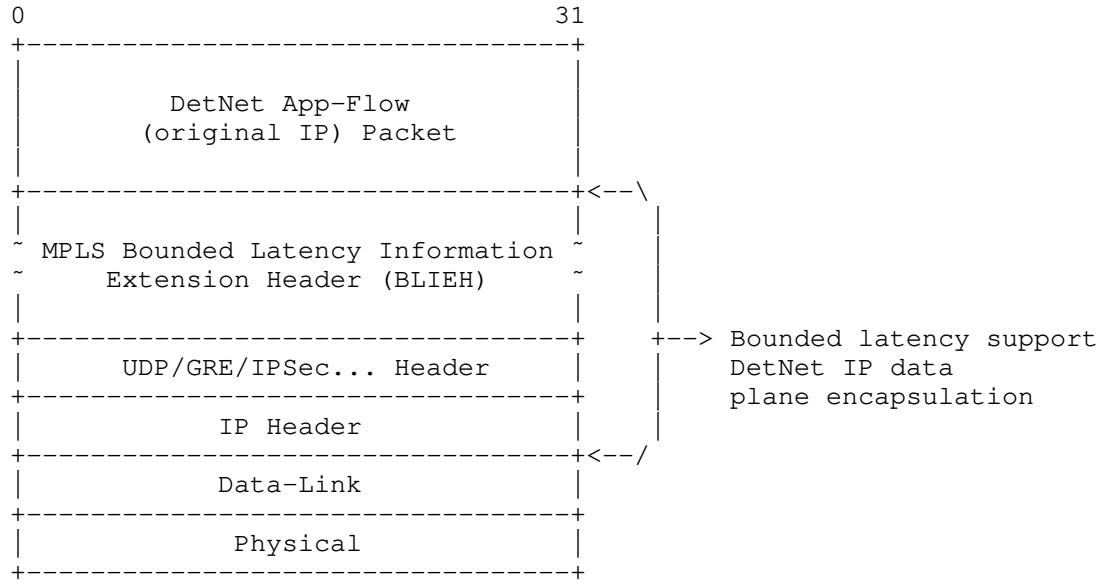


Figure 8: IPv6 extension option of bounded latency

6. IANA Considerations

6.1. New Destination Options and Hop-by-Hop Options

IANA is requested to allocate a value of "Destination Options and Hop-by-Hop Options" under "Internet Protocol Version 6 (IPv6) Parameters" registry. The suggested value is:

Hex	act	chg	rest	Description	Reference
TBD	00	1	TBD	BLI Option	This I-D

Bounded Latency Information Option

6.2. New Type of MPLS Extension Header

IANA is requested to allocate a 8-bit indicator for the Next Header to the Bounded Latency Extension Header.

6.3. New Subregistry of Bounded Latency Information Type

IANA is requested to define a new subregistry of "Bounded Latency Information Type" for the "Bounded Latency Information Option" under "Internet Protocol Version 6 (IPv6) Parameters" registry.

This new subregistry will include the following registries:

Suggested Value	Meaning	Reference
TBD	Reserved	This I-D
TBD	Time resource ID	This I-D
TBD	Priority	This I-D
TBD	End-to-end delay budget	This I-D
TBD	Local delay budget	This I-D
TBD	End-to-end deadline	This I-D
TBD	Local deadline	This I-D
TBD	End-to-end delay variation budget	This I-D
TBD	Local delay variation budget	This I-D

Bounded Latency Information Type

7. Security Considerations

TBD

8. References

8.1. Normative References

[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.ietf-detnet-yang]
Geng, X., Ryoo, Y., Fedyk, D., Rahman, R., and Z. Li, "Deterministic Networking (DetNet) YANG Model", Work in Progress, Internet-Draft, draft-ietf-detnet-yang-17, 4 October 2022, <https://www.ietf.org/archive/id/draft-ietf-detnet-yang-17.txt>.

[I-D.liu-detnet-large-scale-requirements]
Liu, P., Li, Y., Eckert, T., Xiong, Q., Ryoo, J., Zhu, S., and X. Geng, "Requirements for Large-Scale Deterministic Networks", Work in Progress, Internet-Draft, draft-liu-detnet-large-scale-requirements-05, 20 October 2022, <https://datatracker.ietf.org/api/v1/doc/document/draft-liu-detnet-large-scale-requirements/>.

[I-D.mohammadpour-detnet-bounded-delay-variation]
Mohammadpour, E. and J. Le Boudec, "DetNet Bounded Packet-Delay-Variation", Work in Progress, Internet-Draft, draft-mohammadpour-detnet-bounded-delay-variation-00, 10 September 2021, <https://www.ietf.org/archive/id/draft-mohammadpour-detnet-bounded-delay-variation-00.txt>.

[I-D.song-mpls-extension-header]
Song, H., Zhou, T., Andersson, L., Zhang, Z. J., and R. Gandhi, "MPLS Network Actions using Post-Stack Extension Headers", Work in Progress, Internet-Draft, draft-song-mpls-extension-header-11, 15 October 2022, <https://www.ietf.org/archive/id/draft-song-mpls-extension-header-11.txt>.

[I-D.stein-srtsn]
Stein, Y. J., "Segment Routed Time Sensitive Networking", Work in Progress, Internet-Draft, draft-stein-srtsn-01, 29 August 2021, <https://www.ietf.org/archive/id/draft-stein-srtsn-01.txt>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[RFC8939] Varga, B., Ed., Farkas, J., Berger, L., Fedyk, D., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP", RFC 8939, DOI 10.17487/RFC8939, November 2020, <https://www.rfc-editor.org/info/rfc8939>.

[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>.

[RFC9023] Varga, B., Ed., Farkas, J., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP over IEEE 802.1 Time-Sensitive Networking (TSN)", RFC 9023, DOI 10.17487/RFC9023, June 2021, <https://www.rfc-editor.org/info/rfc9023>.

[RFC9024] Varga, B., Ed., Farkas, J., Malis, A., Bryant, S., and D. Fedyk, "Deterministic Networking (DetNet) Data Plane: IEEE 802.1 Time-Sensitive Networking over MPLS", RFC 9024, DOI 10.17487/RFC9024, June 2021, <https://www.rfc-editor.org/info/rfc9024>.

[RFC9025] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: MPLS over UDP/IP", RFC 9025, DOI 10.17487/RFC9025, April 2021, <https://www.rfc-editor.org/info/rfc9025>.

[RFC9030] Thubert, P., Ed., "An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)", RFC 9030, DOI 10.17487/RFC9030, May 2021, <https://www.rfc-editor.org/info/rfc9030>.

[RFC9034] Thomas, L., Anamalamudi, S., Anand, S.V.R., Hegde, M., and C. Perkins, "Packet Delivery Deadline Time in the Routing Header for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 9034, DOI 10.17487/RFC9034, June 2021, <https://www.rfc-editor.org/info/rfc9034>.

[RFC9037] Varga, B., Ed., Farkas, J., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: MPLS over IEEE 802.1 Time-Sensitive Networking (TSN)", RFC 9037, DOI 10.17487/RFC9037, June 2021, <https://www.rfc-editor.org/info/rfc9037>.

[RFC9056] Varga, B., Ed., Berger, L., Fedyk, D., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: IP over MPLS", RFC 9056, DOI 10.17487/RFC9056, October 2021, <https://www.rfc-editor.org/info/rfc9056>.

8.2. Informative References

[I-D.peng-detnet-deadline-based-forwarding] Peng, S., Tan, B., and P. Liu, "Deadline Based Deterministic Forwarding", Work in Progress, Internet-Draft, draft-peng-detnet-deadline-based-forwarding-04, 8 December 2022, <https://www.ietf.org/archive/id/draft-peng-detnet-deadline-based-forwarding-04.txt>.

[I-D.yizhou-detnet-ipv6-options-for-cqf-variant] Li, Y., Ren, S., Li, G., Yang, F., Ryoo, J., and P. Liu, "IPv6 Options for Cyclic Queuing and Forwarding Variants", Work in Progress, Internet-Draft, draft-yizhou-detnet-ipv6-options-for-cqf-variant-01, 24 October 2022, <https://www.ietf.org/archive/id/draft-yizhou-detnet-ipv6-options-for-cqf-variant-01.txt>.

Appendix A. BLI Examples

The following examples are provided to give instructions on how Bounded Latency Information is used when network node implements different algorithms to guarantee the bounded latency transmission.

A.1. Cycle Based Algorithms

When network node implements cycle based algorithms for example [I-D.yizhou-detnet-ipv6-options-for-cqf-variant] , cycles are the local resources used to guarantee the bounded latency transmission. Cycle ID is expected to be carried in data plane. Thus, the data field of BLI is suggested as follows:

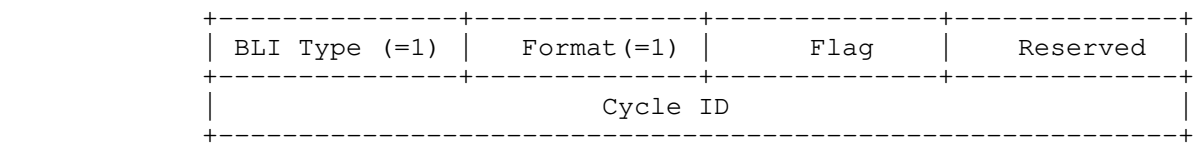


Figure A.1: Data Field of BLI Used With Cycle Based Algorithms

A.2. Time Slot Based Algorithms

When network node implements time slot based algorithms, time slots are the local resources used to guarantee the bounded latency transmission. Time Slot ID is expected to be carried in data plane. Thus, the data field of BLI is suggested as follows:

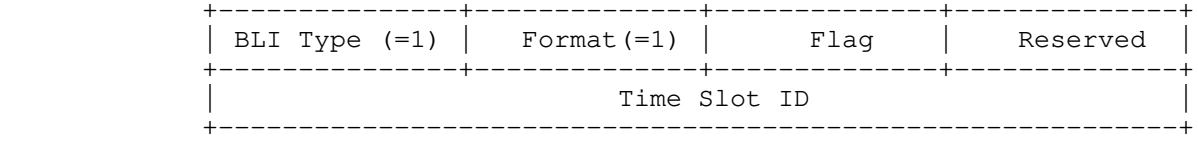


Figure A.2: Data Field of BLI Used With Time Slot Based Algorithms

A.3. Budget Based Algorithms

When network node implements the budget based algorithms to provide bounded latency transmission, end to end or per hop delay budget or delay variation budget information is the requirement proposed from the services and expected to be carried in data plane. The data fields of BLI used with delay budget based algorithms are suggested as follows:

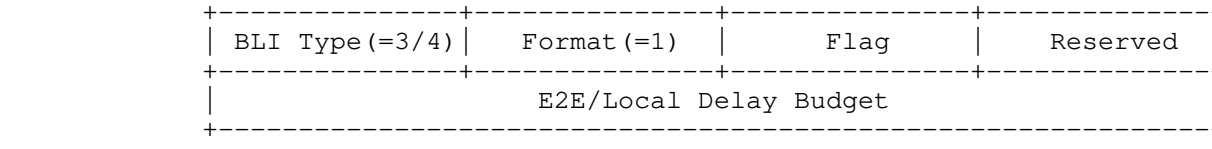


Figure A.3: Data Field of BLI Used With Delay Budget Based Algorithms

The data fields of BLI used with delay variation budget based algorithms are suggested as follows:

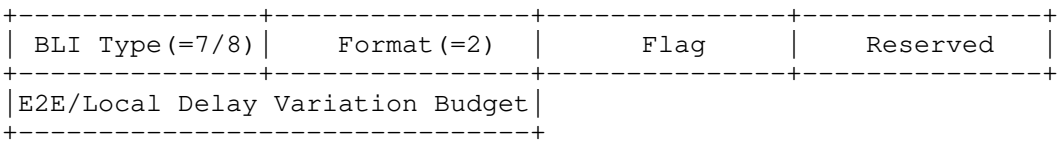


Figure A.4: Data Field of BLI Used With Delay Variation Budget Based Algorithms

A.4. Deadline Based Algorithms

When network node implements deadline based algorithms like EDF, Deadine forwarding [I-D.peng-detnet-deadline-based-forwarding] to provide bounded latency transmission, end to end or per hop packet deadline is the requirement proposed from the services and expected to be carried in data plane. The data fields of BLI used with deadline based algorithms are suggested as follows:

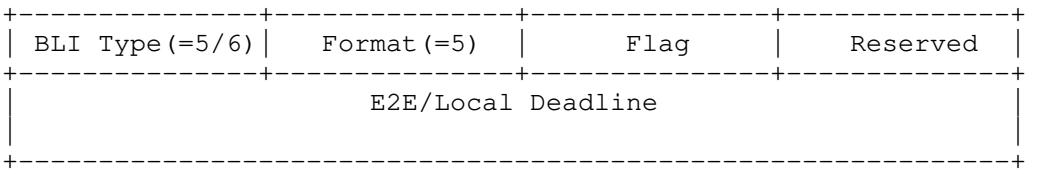


Figure A.5: Data Field of BLI Used With Deadline Based Algorithms

A.5. Priority Based Algorithms

When network node implements priority based algorithms, priority is the requirement proposed from the services. Priority ID is expected to be carried in data plane. The data field of BLI is suggested as follows:

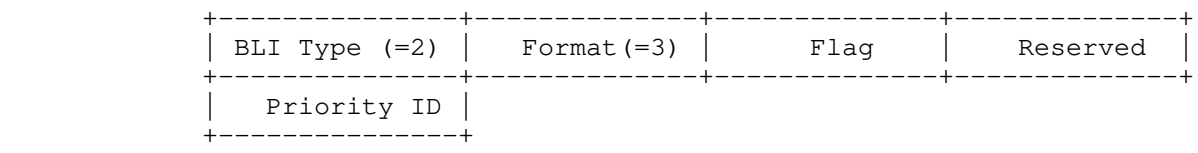


Figure A.6: Data Field of BLI Used With Priority Based Algorithms

Authors' Addresses

Xuesong Geng
Huawei
156 Beiqing Rd.
Beijing
100095
China
Email: gengxuesong@huawei.com

Tianran Zhou
Huawei
156 Beiqing Rd.
Beijing
100095
China
Email: zhoutianran@huawei.com

Li Zhang
Huawei
156 Beiqing Rd.
Beijing
100095
China
Email: zhangli344@huawei.com

Zongpeng Du
China Mobile
No.32 XuanWuMen West Street
Beijing
100053
China
Email: duzongpeng@foxmail.com