Abstract

This document addresses the multi-domain DetNet problem, analyzing what the technical gaps are and exploring some possible solutions. Application, control and data plane aspects are in scope. The goal is to help understanding what might be the next steps towards enabling DetNet in multi technology and/or administrative domains.

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 January 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.
1. Introduction and Problem Statement

The Deterministic Networking (DetNet) Working Group focuses on deterministic data paths that operate over Layer 2 bridged and Layer 3 routed segments, where such paths can provide bounds on latency, loss, and packet delay variation (jitter), and high reliability.

The DetNet architecture document [RFC8655] includes the concept of multi-domain in the DetNet Service reference model (Fig. 5 of [RFC8655], reproduced here in Figure 1 for convenience. However, the WG has not yet worked in detail on the necessary protocol operations to support multi-domain at control and data plane.
Figure 1: DetNet Service Reference Model (Multidomain) (from RFC8655)
In addition to the DetNet work, there is also wireless-focused efforts being explored at the Reliable and Available Wireless (RAW) WG. Wireless operates on a shared medium, and transmissions cannot be fully deterministic due to uncontrolled interferences, including self-induced multipath fading. RAW is an effort to provide Deterministic Networking on across a path that include a wireless interface. RAW provides for high reliability and availability for IP connectivity over a wireless medium. The wireless medium presents significant challenges to achieve deterministic properties such as low packet error rate, bounded consecutive losses, and bounded latency. RAW extends the DetNet Working Group concepts to provide for high reliability and availability for an IP network utilizing scheduled wireless segments and other media, e.g., frequency/time-sharing physical media resources with stochastic traffic: IEEE Std. 802.15.4 timeslotted channel hopping (TSCH), 3GPP 5G ultra-reliable low latency communications (URLLC), IEEE 802.11ax/be, and L-band Digital Aeronautical Communications System (LDACS), etc. Similar to DetNet, RAW technologies aim at staying abstract to the radio layers underneath, addressing the Layer 3 aspects in support of applications requiring high reliability and availability.

DetNet defines the Packet Replication, Elimination, and Ordering Functions (PREOF) as a way to provide service protection. PREOF involves 4 capabilities:

* Sequencing information, by adding a sequence number or time stamp as part of DetNet. This is typically done once, at or near the source.

* Replicating packets into multiple DetNet member flows, and typically sending them along multiple different paths to the destination(s).

* Eliminating duplicate packets of a DetNet flow based on the sequencing information and a history of received packets.

* Reordering DetNet flow’s packets that are received out of order.

Packet (hybrid) ARQ, Replication, Elimination and Ordering (PAREO) is a superset of DetNet’s PREOF, defined in RAW, that includes radio-specific techniques such as short range broadcast, MUMIMO, constructive interference and overhearing, which can be leveraged separately or combined to increase the reliability.

There multiple scenarios and use cases that might involve multiple technology and/or administrative domains in DetNet and RAW. For example, there are several use cases [I-D.ietf-raw-use-cases] where reliability and availability are key requirements for wireless
heterogeneous networks. A couple of relevant examples are (i) the manufacturing sector, where a plethora of devices are interconnected and generate data that need to be reliably delivered to the control and monitoring agents; and (ii) the residential gaming, with eXtended Reality (XR).

Next sections explore what the main multi-domain aspects for the application, controller and network/data planes in DetNet and RAW are, to then identify some existing gaps that would require further work at the IETF.

2. Application plane

As described in [RFC8655], the Application Plane incorporates the User Agent, a specialized application that interacts with the end user and operator and performs requests for DetNet services via an abstract Flow Management Entity (FME), which may or may not be collocated with (one of) the end systems. At the Application Plane, a management interface enables the negotiation of flows between end systems.

In a multi-domain deployment, the User Agent might be aware of the existence of multiple domains or it might be unaware. A multi-domain aware User Agent/application plane could take care of the negotiation of the flows at all involved domains, whereas a multi-domain unaware one will have to rely on the network to take care of it transparently.

3. Controller plane

We refer to the controller plane as the aggregation of the Control and Management planes. The term "Controller Plane Function (CPF)" refers to any device operating in that plane, whether it is a Path Computation Element (PCE) [RFC4655], a Network Management Entity (NME), or a distributed control protocol. The CPF is a core element of a controller, in charge of computing deterministic paths to be applied in the Network Plane. A (Northbound) Service Interface enables applications in the Application Plane to communicate with the entities in the Controller Plane.
In DetNet, one or more CPFs collaborate to implement the requests from the FME as per-flow, per-hop behaviors installed in the DetNet nodes for each individual flow. Adding multi-domain support might require some support at the CPF. For example, CPFs sitting at different domains need to discover themselves, authenticate and negotiate per-hop behaviors. Depending on the multi-domain support provided by the application plane, the controller plane might be relieved from some responsibilities (e.g., if the application plane is taking care of splitting what needs to be provided by each domain).

Let’s know take the case of RAW. As introduced in [I-D.ietf-raw-architecture], RAW separates the path computation time scale at which a complex path is recomputed from the path selection time scale at which the forwarding decision is taken for one or a few packets. RAW operates at the path selection time scale. The RAW problem is to decide, amongst the redundant solutions that are proposed by the Patch Computation Element (PCE), which one will be used for each packet to provide a Reliable and Available service while minimizing the waste of constrained resources. To that effect, RAW defines the Path Selection Engine (PSE) that is the counter-part of the PCE to perform rapid local adjustments of the forwarding tables within the diversity that the PCE has selected for the Track. The PSE enables to exploit the richer forwarding capabilities with Packet (hybrid) ARQ, Replication, Elimination and Ordering (PAREO), and scheduled transmissions at a faster time scale.

While there exist inter-PCE solutions today, allowing one domain’s PCE to learn some inter-domain paths, this would not be sufficient, as the PSE of one domain would not have full visibility nor capability to act on the other domains (e.g., there are no multi-domain OAM solutions in place yet), limiting its capability to guarantee any given SLA. Therefore, there is a need to define inter-PSE coordination mechanisms across domains.

There exist today standardized solutions, such as the ones in the context of Path Computation Element (PCE), enabling computing multi-/inter-domain paths. As an example, the Hierarchical PCE (G-PCE) was defined in RFC 6805 [RFC6805] and is described hereafter. A parent PCE maintains a domain topology map that contains the child domains (seen as vertices in the topology) and their interconnections (links in the topology). The parent PCE has no information about the content of the child domains; that is, the parent PCE does not know about the resource availability within the child domains, nor does it know about the availability of connectivity across each domain because such knowledge would violate the confidentiality requirement and either would require flooding of full information to the parent (scaling issue) or would necessitate some form of aggregation. The parent PCE is used to compute a multi-domain path based on the domain
connectivity information. A child PCE may be responsible for single or multiple domains and is used to compute the intra-domain path based on its own domain topology information.

Solutions like the above are not sufficient alone to solve the multi-domain RAW problem, as the PSEs need to have some additional information from the other involved domains to be sensitive/reactive to transient changes, in order to ensure a certain level of reliability and availability in a multi-domain wireless heterogeneous mesh network. [I-D.bernardos-raw-multidomain] explores in more detail the RAW-specific multi-domain problem and proposes some initial solutions.

4. Network/Data plane

The Network Plane represents the network devices and protocols as a whole, regardless of the layer at which the network devices operate. It includes the Data Plane and Operational Plane (e.g., OAM) aspects. A Southbound (Network) Interface enables the entities in the Controller Plane to communicate with devices in the Network Plane.

At the Network Plane, DetNet nodes may exchange information regarding the state of the paths, between adjacent DetNet nodes and possibly with the end systems. In a multi-domain environment, nodes belonging to different domains might need to exchange information. This might require protocol translations and/or abstractions, as the different domains might not offer the same capabilities nor use the same network protocols. Additionally, OAM protocols [I-D.ietf-detnet-oam-framework] might also need to be extended to support multi-domain operation.

Note as well, that performing PREOF or PAREO across multiple domains poses additional challenges, as knowledge of all the involved domains might not be available and/or the data planes at each domain could also be different.

5. Requirements

TBD.

6. IANA Considerations

TBD.

7. Security Considerations

TBD.
8. Acknowledgments

This work has been partially supported by the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D 6G-DATADRIVEN-04.

9. Informative References

[I-D.bernardos-raw-multidomain]

[I-D.ietf-detnet-oam-framework]

[I-D.ietf-raw-architecture]

[I-D.ietf-raw-use-cases]


Authors’ Addresses

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
28911 Leganes, Madrid
Spain
Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Alain Mourad
InterDigital Europe
Email: Alain.Mourad@InterDigital.com
URI: http://www.InterDigital.com/
Abstract

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

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.
1.  Introduction (informative) ........................................ 2

2.  Using TCQF in the DetNet Architecture and MPLS forwarding plane (informative) ............................................. 3

3.  TCQF per-flow stateless forwarding (normative) ................. 6
  3.1.  Configuration Data model and tag processing for MPLS TC tags ......................................................... 6
  3.2.  Packet processing .................................................. 6
  3.3.  TCQF with label stack operations .............................. 8
  3.4.  TCQF Pseudocode (normative) ................................. 8

4.  TCQF Per-flow Ingress forwarding (normative) .................... 9
  4.1.  Ingress Flows Configuration Data Model ..................... 10
  4.2.  Ingress Flows Pseudocode ...................................... 10

5.  Implementation, Deployment, Operations and Validation considerations (informative) ........................................ 11
  5.1.  High-Speed Implementation .................................... 11
  5.2.  Controller plane computation of cycle mappings .......... 12
  5.3.  Link speed and bandwidth sharing ............................ 14
  5.4.  Validation ....................................................... 14

6.  Security Considerations ............................................ 14

7.  IANA Considerations ................................................ 14

8.  Changelog .......................................................... 15

9.  References .......................................................... 15
  9.1.  Normative References ........................................... 15
  9.2.  Informative References ......................................... 16

Authors’ Addresses ..................................................... 17

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 a cycle identifier, and a limited number of cycles, e.g.: 3...7 are used to overcome these distance and clock synchronization limitations. Because this memo defines how to use the TC field of MPLS LSE as the tag to carry the cycle identifier, it calls this scheme TC Tagged multiple buffer CQF (TC TCQF). See [I-D.qiang-DetNet-large-scale-DetNet] and [I-D.dang-queuing-with-multiple-cyclic-buffers] for more details of the theory of operations of TCQF. Note that TCQF is not necessarily limited to deterministic operations but could also be used in conjunction with congestion controlled traffic, but those considerations are outside the scope of this memo.

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

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

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

For bounded latency QoS mechanisms relying on per-flow regulator state, such as in [TSN-ATS], this requires the use of a per-detnet flow F-Label across the network from S-PE1 to S-PE2, for example through RSVP-TE [RFC3209] enhanced as necessary with QoS parameters matching the underlying bounded latency mechanism (such as [TSN-ATS]).
With TC TCQF, a sequence of LSR and DetNet service node implements TC TCQF, ideally from T-PE1 (ingress) to T-PE2 (egress). The ingress node needs to perform per-DetNet-flow per-packet "shaping" to assign each packet of a flow to a particular TCQF cycle. This ingress-edge-function is currently out of scope of this document (TBD), but would be based on the same type of edge function as used in CQF.

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

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

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

A service node must act as an egress/ingress edge of a TCQF domain if it needs to perform operations that do change the timing of packets other than the type of latency that can be considered in configuration of TCQF (see Section 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 TCQF edge node as long as it does not introduce latencies not included in the TCQF setup and the controller plane reserves resources for the multitude of flows created by the replication taking the allocation of resources in the TCQF cycles into account.

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

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

This configuration model is extensible for interfaces with other tagging, such as IP/DSCP in other documents.

# 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]

Figure 2: TCQF Configuration Data Model

3.2. Packet processing

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

tcqf contains the router/LSR wide configuration of TCQF parameters, independent of the specific tagging mechanism on any interface. Any interface can have a different tagging method.

The model represents a single TCQF domain, which is a set of interfaces acting both as ingress (iif) and egress (oif) interfaces, capable to forward TCQF packets amongst each other. A router/LSR 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. router/LSR 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. router/LSR 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:

\[
\text{tcqf.cycle_clock_offset} = (\text{clock1 mod } (\text{tcqf.cycle_time } \times \text{tcqf.cycles})
\]

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. When tcqf_tc[oif] is configured, oif will use MPLS TC tagging for TCQF. This mapping 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.
3.3. TCQF with 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, 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 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.

```c
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 // other future encap/tagging options for TCQF
    }
  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
    tcqf.if_config[oif]) { // TCQF enabled
    // Map tcqf_cycle iif to oif
```
cycle = pak.context.tcqf_cycle
    = map_cycle(cycle,
        tcqf.if_config[oif].cycle_map[iif])

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

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

// Started when TCQF is enabled on an interface
// dequeues packets from oif.cycleq
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 DetNet packets is performed via
ingres_flow_enqueue(pak) and ingres_flow_2_tcqf(oif,cycle) as
explained in Section 4.2.

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

```plaintext
# 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.

```plaintext
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.

```c
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 requires Time-Gate 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.

$$B[f] \leq \text{MaxPacketsPerInterval} \times \text{MaxPayloadSize} \times 8$$

$$\text{maxcycles} = \max(\text{ceil}(\frac{B[f]}{\text{tcqf.iflow[f].csize}}) | f)$$

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.
Consider in Figure 7 that Router R1 sends packets via C = 3 cycles with a cycle_clock offset of 01 towards Router R2. These packets arrive at R2 with a cycle_clock offset of 01' which includes through D all latencies incurred between releasing a packet on R1 from the cycle buffer until it can be put into a cycle buffer on R2: serialization delay on R1, link delay, non_CQF delays in R1 and R2, especially forwarding in R2, potentially across an internal fabric to the output interface with the sending cycle buffers.

$$A = \left( \text{ceil}\left( \frac{(01' - O2)}{CT}\right) + C + 1 \right) \mod CC$$

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

Figure 8: Calculating cycle mapping

In general, D will be variable \([D_{\text{min}}...D_{\text{max}}]\), 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

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.

9. References

9.1. Normative References

[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,

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Parkas,
"Deterministic Networking Architecture", RFC 8655,
DOI 10.17487/RFC8655, October 2019,

Eckert, et al. Expires 12 January 2023
9.2. Informative References

[I-D.dang-queuing-with-multiple-cyclic-buffers]

[I-D.eckert-detnet-bounded-latency-problems]

[I-D.ietf-bier-te-arch]

[I-D.ietf-detnet-bounded-latency]

[I-D.qiang-DetNet-large-scale-DetNet]
[IEEE802.1Qbv]

[IEEE802.1Qch]

[LDN]

[RFC3209]

[RFC8402]

[RFC9016]

[TSN-ATS]

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
Abstract

This document provides a framework overview for the Deterministic Networking (DetNet) controller plane. It discusses concepts and requirements for DetNet controller plane which could be basis for future solution specification.

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 31 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 ............................................. 3
  1.1. Terminology ........................................ 4
2. DetNet Controller Plane Requirements .......................... 4
  2.1. DetNet Control Plane Requirements ...................... 4
  2.2. DetNet Management Plane Requirements ................... 5
  2.3. Requirements For Both Planes .......................... 5
3. DetNet Control Plane Architecture ............................ 6
  3.1. Distributed Control Plane and Signaling Protocols ........ 6
  3.2. SDN/Fully Centralized Control Plane ................... 7
  3.3. Hybrid Control Plane (partly centralized, partly distributed) ............................................. 7
  4.1. Explicit Paths ........................................ 8
  4.2. Resource Reservation .................................. 8
  4.3. PREOF Support ......................................... 9
  4.4. Data Plane specific considerations ..................... 9
    4.4.1. DetNet in an MPLS Domain .......................... 9
    4.4.2. DetNet in an IP Domain ............................ 10
    4.4.3. DetNet in a Segment Routing Domain ................. 11
5. Management Plane Overview ................................... 11
  5.1. Provisioning .......................................... 11
  5.2. DetNet Operations, Administration and Maintenance
        (OAM) ................................................ 12
    5.2.1. OAM for Performance Monitoring (PM) ................ 12
    5.2.2. OAM for Connectivity and Fault/Defect Management
            (CFM) ............................................ 12
6. Gap Analysis ............................................... 12
7. IANA Considerations ........................................ 12
8. Security Considerations ..................................... 13
9. Acknowledgments ............................................ 13
10. References ............................................... 13
    10.1. Normative References ............................... 13
    10.2. Informative References .............................. 16
Authors’ Addresses ............................................ 18
1. Introduction

Deterministic Networking (DetNet) provides the capability to carry specified unicast and/or multicast data flows for real-time applications with extremely low data loss rates and bounded latency within a network domain. As defined in [RFC8655], techniques used to provide DetNet capability include reserving data plane resources for individual (or aggregated) DetNet flows in some or all of the intermediate nodes along the path of the flow, providing explicit routes for DetNet flows that do not immediately change with the network topology, and distributing data from DetNet flow packets over time and/or space to ensure delivery of each packet’s data in spite of the loss of a path.

DetNet data plane is defined in a set of documents that are anchored by the DetNet Data Plane Framework [RFC8938] (and the associated DetNet MPLS defined in [RFC8964] and DetNet IP defined in [RFC8939] and other data plane specifications defined in [RFC9023], [RFC9024], [RFC9025], [RFC9037] and [RFC9056]).

While the Detnet Architecture and Data Plane documents are primarily concerned with data plane operations, they do contain some requirements for functions that would be required in order to automate DetNet service provisioning and monitoring via a DetNet controller plane. The purpose of this document is to gather these requirements into a single document and discuss how various possible DetNet controller plane architectures could be used to satisfy these requirements, while not providing the protocol details for a DetNet controller plane solution. Such controller plane protocol solutions will be the subject of subsequent documents.

Note that in the DetNet overall architecture, the controller plane includes what are more traditionally considered separate control and management planes. Traditionally, the management plane is primarily involved with fault management, configuration management and performance management (sometimes accounting management and security management is also considered in the management plane, but not in the scope of this document)., while the control plane is primarily responsible for the instantiation and maintenance of flows, MPLS label allocation and distribution, and active in-band or out-of-band signaling to support DetNet functions. In the DetNet architecture, all of this functionality is combined into a single Controller Plane. See Section 4.4.2 of [RFC8655] and the aggregation of Control and Management planes in [RFC7426] for further details.
1.1. Terminology

This document uses the terminology established in the DetNet Architecture [RFC8655], and the reader is assumed to be familiar with that document and its terminology.

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] when, and only when, they appear in all capitals, as shown here.

2. DetNet Controller Plane Requirements

Other DetNet documents, including [RFC8655] and [RFC8938], contain requirements for the Controller Plane. For convenience, these requirements have been compiled here. These requirements have been organized into 3 groups, including: requirements primarily applicable to control plane, requirements primarily applicable to management plane and requirements applicable to both planes.

2.1. DetNet Control Plane Requirements

The primary requirements for the DetNet Control Plane include:

* Support the dynamic creation, modification, and deletion of DetNet flows. This may include some or all of explicit path determination, link bandwidth reservations, restricting flows to specific links (e.g., IEEE 802.1 Time-Sensitive Networking (TSN) links), node buffer and other resource reservations, specification of required queuing disciplines along the path, ability to manage bidirectional flows, etc., as needed for a flow.

* Support DetNet flow aggregation and de-aggregation via the ability to dynamically create and delete flow aggregates (FAs), and be able to modify existing FAs by adding or deleting participating flows.

* Allow flow instantiation requests to originate in an end application (via an Application Programming Interface (API), via static provisioning, or via a dynamic control plane, such as a centralized SDN controller or distributed signaling protocols. See Section 3 for further discussion of these options.

* In the case of the DetNet MPLS data plane, manage DetNet Service Label (S-Label), Forwarding Label (F-Label), and Aggregation Label (A-Label) [RFC8964] allocation and distribution.
Also in the case of the DetNet MPLS data plane, support the DetNet service sub-layer, which provides DetNet service functions such as protection and reordering through the use of packet replication, duplicate elimination, and packet ordering functions (PREOF).

Support queue control techniques defined in Section 4.5 of [RFC8655] and [I-D.ietf-detnet-bounded-latency] that require time synchronization among network nodes.

Advertise static and dynamic node and link resources such as capabilities and adjacencies to other network nodes (for dynamic signaling approaches) or to network controllers (for centralized approaches).

Scale to handle the number of DetNet flows expected in a domain (which may require per-flow signaling or provisioning).

Provision flow identification information at each of the nodes along the path. Flow identification may differ depending on the location in the network and the DetNet functionality (e.g. transit node vs. relay node).

2.2. DetNet Management Plane Requirements

The primary requirements of the DetNet Management Plane are that it must be able to:

Monitor the performance of DetNet flows and nodes to ensure that they are meeting required objectives, both proactively and on-demand.

Support DetNet flow continuity check and connectivity verification functions.

Support testing and monitoring of packet replication, duplicate elimination, and packet ordering functionality in the DetNet domain.

2.3. Requirements For Both Planes

The following requirements apply to both the DetNet Controller and Management Planes:

Operate in a converged network domain that contains both DetNet and non-DetNet flows.

Adapt to DetNet domain topology changes such as links or nodes failures (fault recovery/restoration), additions and removals.
3. DetNet Control Plane Architecture

As noted in the Introduction, the DetNet control plane is responsible for the instantiation and maintenance of flows, allocation and distribution of flow related information (e.g., MPLS label), and active in-band or out-of-band information distribution to support these functions.

The following sections define three types of DetNet control plane architectures: a fully distributed control plane utilizing dynamic signaling protocols, a fully centralized SDN-like control plane, and a hybrid control plane containing both distributed protocols and centralized controlling. This document describes the various information exchanges between entities in the network for each type of these architectures and the corresponding advantages and disadvantages.

In each of the following sections, there are examples to illustrate possible mechanisms that could be used in each type of the architectures. They are not meant to be exhaustive or to preclude any other possible mechanism that could be used in place of those used in the examples.

3.1. Distributed Control Plane and Signaling Protocols

In a fully distributed configuration model, User-to-Network Interface (UNI) information is transmitted over a DetNet UNI protocol from the user side to the network side. Then UNI and network configuration information propagate in the network via distributed control plane signaling protocols. Such a DetNet UNI protocol is not necessary in case that the End-systems are DetNet capable.

Taking an RSVP-TE MPLS network as an example, where end systems are not part of the DetNet domain:

1. Network nodes collects topology information and DetNet capabilities of the network nodes through IGP;

2. Ingress edge node receives a flow establishment request from the UNI and calculates one or more valid path(s);

3. The ingress node sends a PATH message with an explicit route through RSVP-TE [RFC3209]. After receiving the PATH message, the egress edge node sends a RESV message with the distributed label and resource reservation request.

In this example, both IGP and RSVP-TE may request extensions for DetNet.
3.2. SDN/Fully Centralized Control Plane

In the fully SDN/centralized configuration model, flow/UNI information is transmitted from a Centralized User Controller or from applications via an API/ northbound interface to a Centralized Controller. Network node configurations for DetNet flows are performed by the controller using a protocol such as NETCONF [RFC6241]/YANG [RFC6020] or PCE-CC [RFC8283].

Take the following case as an example:

1. A Centralized Controller collects topology information and DetNet capabilities of the network nodes via NETCONF/YANG;
2. The Controller receives a flow establishment request from a UNI and calculates one or more valid path(s) through the network;
3. The Controller chooses the optimal path and configures the devices along that path for DetNet flow transmission via PCE-CC.

Protocols in the above example may require extensions for DetNet.

3.3. Hybrid Control Plane (partly centralized, partly distributed)

In the hybrid model, controller and control plane protocols work together to provide DetNet services, and there are a number of possible combinations.

In the following case, RSVP-TE and controller are used together:

1. Controller collects topology information and DetNet capabilities of the network nodes via an IGP and/or BGP-LS [RFC7752];
2. Controller receives a flow establishment request through API and calculates one or more valid path(s) through the network;
3. Based on the calculation result, the Controller distributes flow path information to the ingress edge node and configures network nodes along the path with necessary DetNet information (e.g. for replication/duplicate elimination);
4. Using RSVP-TE, the ingress edge node sends a PATH message with an explicit route. After receiving the PATH message, the egress edge node sends a RESV message with the distributed label and resource reservation request.
There are many other variations that could be included in a hybrid control plane. The requested DetNet extensions for protocol in each possible case is for future work.


This section discusses requested control plane features for DetNet mechanisms as defined in [RFC8655], including explicit path, resource reservation, service protection(PREOF). Different DetNet service may implement part/all of them based on the requirements.

4.1. Explicit Paths

Explicit paths are required in DetNet to provide a stable forwarding service and guarantee that DetNet service is not impacted when the network topology changes. The following features are necessary in control plane to implement explicit paths in DetNet:

* Path computation: DetNet explicit paths need to meet the SLA (Service Level Agreement) requirements of the application, which include bandwidth, maximum end-to-end delay, maximum end-to-end delay variation, maximum loss ratio, etc. In a distributed network system, IGP with CSPF (Constrained Shortest Path First) may be used to compute a set of feasible paths for a DetNet service. In a centralized network system, controller can compute paths satisfying the requirements of DetNet based on the network information collected from the DetNet domain.

* Path establishment: The computed path for the DetNet service has to be sent/configured/signaled to the network device, so the corresponding DetNet flow could pass through the network domain following the specified path.

4.2. Resource Reservation

DetNet flows are supposed to be protected from congestion, so sufficient resource reservation for DetNet service could protect service from congestion. There are multiple types of resources in the network that could be allocated to DetNet flows, e.g., packet processing resource, buffer resource, and bandwidth of the output port. The network resource requested by a specified DetNet service is determined by the SLA requirements and network capability.

* Resource Allocation: Port bandwidth is one of the basic attributes of a network device which is easy to obtain or calculate. In current traffic engineering implementations, network resource allocation is synonymous with bandwidth allocation. A DetNet flow is characterized with a traffic specification as defined in
[RFC9016], including attributes such as Interval, Maximum Packets Per Interval, and Maximum Payload Size. The traffic specification describes the worst case, rather than the average case, for the traffic, to ensure that sufficient bandwidth and buffering resources are reserved to satisfy the traffic specification. However, in case of DetNet, resource allocation is more than simple bandwidth reservation. For example, allocation of buffers and required queuing disciplines during forwarding may be required as well. Furthermore, resources must be ensured to execute DetNet service sub-layer functions on the node, such as protection and reordering through the use of packet replication, duplicate elimination, and packet ordering functions (PREOF).

* Device configuration with or without flow discrimination: The resource allocation can be guaranteed by device configuration. For example, an output port bandwidth reservation can be configured as a parameter of queue management and the port scheduling algorithm. When DetNet flows are aggregated, a group of DetNet flows share the allocated resource in the network device. When the DetNet flows are treated independently, the device should maintain a mapping relationship between a DetNet flow and its corresponding resources.

4.3. PREOF Support

DetNet path redundancy is supported via packet replication, duplicate elimination, and packet ordering functions (PREOF). A DetNet flow is replicated and goes through multiple network paths to avoid packet loss caused by device or link failures. In general, current control plane mechanisms that can be used to establish an explicit path, whether distributed or centralized, support point-to-point (P2P) and point-to-multipoint (P2MP) path establishment. PREOF requires the ability to compute and establish a set of multiple paths (e.g., multiple LSP segments in an MPLS network) from the point(s) of packet replication to the point(s) of packet merging and ordering. Mapping of DetNet (member) flows to explicit path segments has to be ensured as well. Protocol extensions will be required to support these new features. Terminology will also be required to refer to this coordinated set of path segments (such as an "LSP graph" in case of DetNet MPLS data plane).

4.4. Data Plane specific considerations

4.4.1. DetNet in an MPLS Domain

For the purposes of this document, "traditional MPLS" is defined as MPLS without the use of segment routing (see Section 4.4.3 for a discussion of MPLS with segment routing) or MPLS-TP [RFC5960].
In traditional MPLS domains, a dynamic control plane using distributed signaling protocols is typically used for the distribution of MPLS labels used for forwarding MPLS packets. The dynamic signaling protocols most commonly used for label distribution are LDP [RFC5036], RSVP-TE, and BGP [RFC8277] (which enables BGP/MPLS-based Layer 3 VPNs [RFC4384] and Layer 2 VPNs [RFC7432]).

Any of these protocols could be used to distribute DetNet Service Labels (S-Labels) and Aggregation Labels (A-Labels) [RFC8964]. As discussed in [RFC8938], S-Labels are similar to other MPLS service labels, such as pseudowire, L3 VPN, and L2 VPN labels, and could be distributed in a similar manner, such as through the use of targeted LDP or BGP. If these were to be used for DetNet, they would require extensions to support DetNet-specific features such as PREOF, aggregation (A-Labels), node resource allocation, and queue placement.

However, as discussed in Section 3.1, distributed signaling protocols may have difficulty meeting DetNet’s scalability requirements. MPLS also allows SDN-like centralized label management and distribution as an alternative to distributed signaling protocols, using protocols such as PCEP and OpenFlow [OPENFLOW].

PCEP, particularly when used as a part of PCE-CC, is a possible candidate protocol to use for centralized management of traditional MPLS-based DetNet domains. However, PCE path calculation algorithms would need to be extended to include the location determination for PREOF nodes in a path, and the means to signal the necessary resource reservation and PREOF function placement information to network nodes. See ((?I-D.ietf-pce-pcep-extension-for-pce-controller)) for further discussion of PCE-CC and PCEP for centralized control of an MPLS domain.

4.4.2. DetNet in an IP Domain

For the purposes of this document, "traditional IP" is defined as IP without the use of segment routing (see Section 4.4.3 for a discussion of IP with segment routing). In a later revision of this document, this section will discuss possible protocol extensions to existing IP routing protocols such as OSPF, IS-IS, and BGP. It should be noted that a DetNet IP data plane [RFC8939] is simpler than a DetNet MPLS data plane [RFC8964], and doesn’t support PREOF, so only one path per flow or flow aggregate is required.
4.4.3. DetNet in a Segment Routing Domain

Segment Routing [RFC8402] is a scalable approach to building network domains that provides explicit routing via source routing encoded in packet headers and it is combined with centralized network control to compute paths through the network. Forwarding paths are distributed with associated policy to network edge nodes for use in packet headers. As such, segment routing can be considered as a new data plane for both MPLS and IP. It reduces the amount of network signaling associated with distributed signaling protocols such as RSVP-TE, and also reduces the amount of state in core nodes compared with that required for traditional MPLS and IP routing, as the state is now in the packets rather than in the routers. This could be useful for DetNet, where a very large number of flows through a network domain are expected, which would otherwise require the instantiation of state for each flow traversing each node in the network. However, further analysis is needed on the expected gain, as DetNet flows may require various type of DetNet specific resources as well.

In a later revision of this document, this section will discuss the impact of DetNet on the Segment Routing Control and Management planes. Note that the DetNet MPLS and IP data planes described in [RFC8964] and [RFC8939] were constructed to be compatible with both types of segment routing, SR-MPLS [RFC8660] and SRv6 [I-D.ietf-6man-segment-routing-header]. However, as of this writing, traffic engineering and resource reservation for segment routing are currently unsolved problems.

Editor’s note: this section may be collapsed to previous sections and listing MPLS segment routing in the MPLS section as one of the possible explicit routing techniques for MPLS, and do the same for IP.

5. Management Plane Overview

The Management Plane includes the ability to statically provision network nodes and to use OAM to monitor DetNet performance and detect outages or other issues at the DetNet layer.

5.1. Provisioning

Static provisioning in a Detnet network nodes will be performed via the use of appropriate YANG models, including [I-D.ietf-detnet-yang] and [I-D.ietf-detnet-topology-yang].
5.2. DetNet Operations, Administration and Maintenance (OAM)

This document covers the general considerations for OAM.

5.2.1. OAM for Performance Monitoring (PM)

5.2.1.1. Active PM

Active PM is performed by injecting OAM packets into the network to estimate the performance of the network by measuring the performance of the OAM packets. Adding extra traffic can affect the delay and throughput performance of the network, and for this reason active PM is not recommended for use in operational DetNet domains. However, it is a useful test tool when commissioning a new network or during troubleshooting.

5.2.1.2. Passive PM

Passive PM monitors the actual service traffic in a network domain in order to measure its performance without having a detrimental affect on the network. As compared to Active PM, Passive PM is much preferred for use in DetNet domains.

5.2.2. OAM for Connectivity and Fault/Defect Management (CFM)

The detailed requirements for connectivity and fault/defect detection and management in DetNet IP domain and DetNet MPLS domain are defined in respectively in [I-D.ietf-detnet-ip-oam] and [I-D.ietf-detnet-mpls-oam].

6. Gap Analysis

In a later revision of this document, this section will contain a gap analysis of existing IETF control and management plane protocols not already discussed elsewhere in this document for their ability (or inability) to satisfy the requirements in Section 2, and discuss possible protocol extensions to existing protocols to fill the gaps, if any.

7. IANA Considerations

This document has no actions for IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.
8. Security Considerations

Editor’s note: This section needs more details.

The overall security considerations of DetNet are discussed in [RFC8655] and [I-D.ietf-detnet-security]. For DetNet networks that make use of Segment Routing (whether SR-MPLS or SRv6), the security considerations in [RFC8402] also apply.

DetNet networks that make use of a centralized controller plane may be threatened by the loss of connectivity (whether accidental or malicious) between the central controller and the network nodes, and/or the spoofing of control messages from the controller to the network nodes. This is important since such networks depend on centralized controllers to calculate flow paths and instantiate flow state in the network nodes. For networks that use both DetNet and Segment Routing with a centralized controller, this would also include the calculation of SID lists and their installation in edge/border routers.

In both cases, such threats may be mitigated through redundant controllers, the use of authentication between the controller(s) and the network nodes, and other mechanisms for protection against DOS attacks. A mechanism for supporting one or more alternative central controllers and the ability to fail over to such an alternative controller will be required.

9. Acknowledgments

Thanks to Jim Guichard, Donald Eastlake, and Stewart Bryant for their review comments.

10. References

10.1. Normative References

[I-D.ietf-detnet-bounded-latency]
[I-D.ietf-detnet-flow-information-model]

[I-D.ietf-detnet-ip-oam]

[I-D.ietf-detnet-mpls-oam]

[I-D.ietf-detnet-security]


[RFC8174] "".


10.2. Informative References


[I-D.ietf-6man-segment-routing-header]

[I-D.ietf-detnet-topology-yang]

[I-D.ietf-detnet-yang]

[IEEE.802.1QBV_2015]


[RFC4384]  Meyer, D., "BGP Communities for Data Collection", BCP 114,  
RFC 4384, DOI 10.17487/RFC4384, February 2006,  

"LDP Specification", RFC 5036, DOI 10.17487/RFC5036,  

Scaling Issues in MPLS-TE Core Networks", RFC 5439,  
DOI 10.17487/RFC5439, February 2009,  

Transport Profile Data Plane Architecture", RFC 5960,  
DOI 10.17487/RFC5960, August 2010,  

the Network Configuration Protocol (NETCONF)", RFC 6020,  
DOI 10.17487/RFC6020, October 2010,  

and A. Bierman, Ed., "Network Configuration Protocol  
(NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011,  

Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based  
Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February  

S. Ray, "North-Bound Distribution of Link-State and  
Traffic Engineering (TE) Information Using BGP", RFC 7752,  
DOI 10.17487/RFC7752, March 2016,  

[RFC8277]  Rosen, E., "Using BGP to Bind MPLS Labels to Address  
Prefixes", RFC 8277, DOI 10.17487/RFC8277, October 2017,  

Architecture for Use of PCE and the PCE Communication  
Protocol (PCEP) in a Network with Central Control",  
RFC 8283, DOI 10.17487/RFC8283, December 2017,  

Authors’ Addresses

Andrew G. Malis
Independent
Email: agmalis@gmail.com

Xuesong Geng
Huawei
Email: gengxuesong@huawei.com

Mach (Guoyi) Chen
Huawei
Email: mach.chen@huawei.com

Fengwei Qin
China Mobile
Email: qinfengwei@chinamobile.com

Balazs Varga
Ericsson
Email: balazs.a.varga@ericsson.com
Operations, Administration and Maintenance (OAM) for Deterministic Networks (DetNet) with IP Data Plane
draft-ietf-detnet-ip-oam-04

Abstract

This document defines the principles for using Operations, Administration, and Maintenance protocols and mechanisms in the Deterministic Networking networks with the IP data plane.

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 August 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 ................................................. 2
2. Conventions used in this document ............................ 3
   2.1. Terminology ............................................. 3
   2.2. Keywords ................................................ 3
3. Active OAM for DetNet Networks with the IP Data Plane ...... 4
   3.1. Active OAM Using DetNet-in-UDP Encapsulation ........... 4
   3.2. Mapping Active OAM and IP DetNet flows ................ 5
   3.3. Active OAM Using GRE-in-UDP Encapsulation .............. 6
4. OAM of DetNet IP Interworking with OAM of non-IP DetNet domains ................................................. 6
5. IANA Considerations .......................................... 6
6. Security Considerations ....................................... 6
7. Acknowledgment ................................................ 6
8. References .................................................... 6
   8.1. Normative References ................................... 6
   8.2. Informational References ............................... 7
Authors’ Addresses ................................................ 8

1. Introduction

[RFC8655] introduces and explains Deterministic Networks (DetNet) architecture.

Operations, Administration and Maintenance (OAM) protocols are used to detect, localize defects in the network, and monitor network performance. Some OAM functions, e.g., failure detection, work in the network proactively, while others, e.g., defect localization, usually performed on-demand. These tasks achieved by a combination of active and hybrid, as defined in [RFC7799], OAM methods.

[I-D.tpmb-detnet-oam-framework] lists the functional requirements toward OAM for DetNet domain. The list can further be used for gap analysis of available OAM tools to identify possible enhancements of existing or whether new OAM tools are required to support proactive and on-demand path monitoring and service validation. Also, the document defines the OAM use principals for the DetNet networks with the IP data plane.
2. Conventions used in this document

2.1. Terminology

The term "DetNet OAM" used in this document interchangeably with longer version "set of OAM protocols, methods and tools for Deterministic Networks".

DetNet Deterministic Networks

DiffServ Differentiated Services

OAM: Operations, Administration and Maintenance

PREF Packet Replication and Elimination Function

POF Packet Ordering Function

RDI Remote Defect Indication

ICMP Internet Control Message Protocol

ACH Associated Channel Header

Underlay Network or Underlay Layer: The network that provides connectivity between the DetNet nodes. MPLS network providing LSP connectivity between DetNet nodes is an example of the underlay layer.

DetNet Node - a node that is an actor in the DetNet domain. DetNet domain edge node and node that performs PREF within the domain are examples of DetNet node.

2.2. Keywords

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. Active OAM for DetNet Networks with the IP Data Plane

OAM protocols and mechanisms act within the data plane of the particular networking layer. And thus it is critical that the data plane encapsulation supports OAM mechanisms in such a way that DetNet OAM packets are in-band with a DetNet flow being monitored, i.e., DetNet OAM test packets follow precisely the same path as DetNet data plane traffic both for unidirectional and bi-directional DetNet paths.

The DetNet data plane encapsulation in a transport network with IP encapsulations specified in Section 6 of [RFC8939]. For the IP underlay network, DetNet flows are identified by the ordered match to the provisioned information set that, among other elements, includes the IP protocol, source port number, destination port number. Active IP OAM protocols like Bidirectional Forwarding Detection (BFD) [RFC5880] or STAMP [RFC8762], use UDP transport and the well-known UDP port numbers as the destination port. Thus a DetNet node MUST be able to associate an IP DetNet flow with the particular test session to ensure that test packets experience the same treatment as the DetNet flow packets.

Most of on-demand failure detection and localization in IP networks is being done by using the Internet Control Message Protocol (ICMP) Echo Request, Echo Reply and the set of defined error messages, e.g., Destination Unreachable, with the more detailed information provided through code points. [RFC0792] and [RFC4443] define the ICMP for IPv4 and IPv6 networks, respectively. Because ICMP is another IP protocol like, for example, UDP, a DetNet node MUST able to associate an ICMP packet generated by the specified IP DetNet node and addressed to the another IP DetNet node with an IP DetNet flow between this pair of endpoints.

3.1. Active OAM Using DetNet-in-UDP Encapsulation

Active OAM in IP DetNet can be realized using DetNet-in-UDP encapsulation. Using DetNet-in-UDP tunnel between IP DetNet nodes ensures that active OAM test packets are fate-sharing with the monitored IP DetNet flow packets. As a result, a test packet shares the tunnel with the IP DetNet flow and shares the fate, statistically speaking, of the IP DetNet flow being monitored.
[I-D.varga-detnet-ip-preof] describes how DetNet with MPLS over UDP/IP data plane [RFC9025] can be used to support Packet Replication, Elimination, and Ordering Functions to potentially lower packet loss, improve the probability of on-time packet delivery and ensure in-order packet delivery in IP DetNet’s service sub-layer. To ensure that an active OAM test packet follows the path of the monitored DetNet flow in the DetNet service sub-layer the encapsulation shown in Figure 1 is used.

![Figure 1: DetNet Associated Channel Header Format](image)

where DetNet ACH is the DetNet Associated Channel Header defined in [I-D.ietf-detnet-mpls-oam].

3.2. Mapping Active OAM and IP DetNet flows

IP OAM protocols that use UDP transport, e.g., BFD and STAMP, can be used to detect failures or performance degradation that affects an IP DetNet flow. When the UDP destination port number used by the OAM protocol is one of the assigned by IANA, then the UDP source port can be used to achieve co-routedness of OAM, and the monitored IP DetNet flow in the multipath environments, e.g., LAG or ECMP. To maximize the accuracy of OAM results in detecting failures and monitoring performance of IP DetNet, test packets should receive the same treatment by the nodes as experienced by the IP DetNet packet. Hence, the DSCP value used for a test packet MUST be mapped to DetNet.
3.3. Active OAM Using GRE-in-UDP Encapsulation

[RFC8086] has defined the method of encapsulating GRE (Generic Routing Encapsulation) headers in UDP. GRE-in-UDP encapsulation can be used for IP DetNet OAM as it eases the task of mapping an OAM test session to a particular IP DetNet flow that is identified by N-tuple. Matching a GRE-in-UDP tunnel to the monitored IP DetNet flow enables the use of Y.1731/G.8013 [ITU-T.1731] as a comprehensive toolset of OAM. The Protocol Type field in GRE header MUST be set to 0x8902 assigned by IANA to IEEE 802.1ag Connectivity Fault Management (CFM) Protocol / ITU-T Recommendation Y.1731. Y.1731/G.8013 supports necessary for IP DetNet OAM functions, i.e., continuity check, one-way packet loss and packet delay measurement.

4. OAM of DetNet IP Interworking with OAM of non-IP DetNet domains

A domain in which IP data plane provides DetNet service could be used in conjunction with a TSN and a DetNet domain with MPLS data plane to deliver end-to-end service. In such scenarios, the ability to detect defects and monitor performance using OAM is essential. [I-D.ietf-detnet-mpls-oam] identified two OAM interworking models - peering and tunneling. Interworking between DetNet domains with IP and MPLS data planes analyzed in Section 6.2 of [I-D.ietf-detnet-mpls-oam]. Also, requirements and recommendations for OAM interworking between a DetNet domain with MPLS data plane and OAM of a TSN equally apply to a DetNet domain with an IP data plane.

5. IANA Considerations

This document does not have any requests for IANA allocation. This section can be deleted before the publication of the draft.

6. Security Considerations

This document describes the applicability of the existing Fault Management and Performance Monitoring IP OAM protocols, and does not raise any security concerns or issues in addition to ones common to networking or already documented for the referenced DetNet and OAM protocols.

7. Acknowledgment

TBA

8. References

8.1. Normative References

[I-D.ietf-detnet-mpls-oam]


8.2. Informational References


Authors’ Addresses

Greg Mirsky
Ericsson
Email: gregimirsky@gmail.com

Mach(Guoyi) Chen
Huawei
Email: mach.chen@huawei.com
David Black  
Dell EMC  
176 South Street  
Hopkinton, MA, 01748  
United States of America  

Email: david.black@dell.com
Abstract

This document defines format and use principals of the Deterministic Network (DetNet) service Associated Channel (ACH) over a DetNet network with the MPLS data plane. The DetNet service ACH can be used to carry test packets of active Operations, Administration, and Maintenance protocols that are used to detect DetNet failures and measure performance metrics.

Status of This Memo

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

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

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

This Internet-Draft will expire on 8 September 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.
1. Introduction

[RFC8655] introduces and explains Deterministic Networks (DetNet) architecture and how the Packet Replication, Elimination, and Ordering functions (PREOF) can be used to ensure low packet drop ratio in DetNet domain.

Operations, Administration and Maintenance (OAM) protocols are used to detect, localize defects in the network, and monitor network performance. Some OAM functions, e.g., failure detection, work in the network proactively, while others, e.g., defect localization, usually performed on-demand. These tasks achieved by a combination of active and hybrid, as defined in [RFC7799], OAM methods.

Also, this document defines format and use principals of the DetNet service Associated Channel over a DetNet network with the MPLS data plane [RFC8964].
2. Conventions used in this document

2.1. Terminology and Acronyms

The term "DetNet OAM" used in this document interchangeably with longer version "set of OAM protocols, methods and tools for Deterministic Networks".

CW Control Word

DetNet Deterministic Networks

d-ACH DetNet Associated Channel Header

d-CW DetNet Control Word

DNH DetNet Header

GAL Generic Associated Channel Label

G-ACh Generic Associated Channel

OAM: Operations, Administration and Maintenance

PREOF Packet Replication, Elimination, and Ordering Functions

PW Pseudowire

RDI Remote Defect Indication

E2E End-to-end

CFM Connectivity Fault Management

BFD Bidirectional Forwarding Detection

TSN Time-Sensitive Network

F-Label A Detnet "forwarding" label that identifies the LSP used to forward a DetNet flow across an MPLS PSN, e.g., a hop-by-hop label used between label switching routers (LSR).

S-Label A DetNet "service" label that is used between DetNet nodes that implement also the DetNet service sub-layer functions. An S-Label is also used to identify a DetNet flow at DetNet service sub-layer.
Underlay Network or Underlay Layer: The network that provides connectivity between the DetNet nodes. MPLS network providing LSP connectivity between DetNet nodes is an example of the underlay layer.

DetNet Node – a node that is an actor in the DetNet domain. DetNet domain edge node and node that performs PREOF within the domain are examples of DetNet node.

2.2. Keywords

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. Active OAM for DetNet Networks with MPLS Data Plane

OAM protocols and mechanisms act within the data plane of the particular networking layer. And thus it is critical that the data plane encapsulation supports OAM mechanisms in such a way to comply with the OAM requirements listed in [I-D.tpmb-detnet-oam-framework]. One of such examples that require special consideration is requirement #5:

DetNet OAM packets MUST be in-band, i.e., follow precisely the same path as DetNet data plane traffic both for unidirectional and bi-directional DetNet paths.

The Det Net data plane encapsulation in transport network with MPLS encapsulation specified in [RFC8964]. For the MPLS underlay network, DetNet flows to be encapsulated analogous to pseudowires (PW) over MPLS packet switched network, as described in [RFC3985], [RFC4385]. Generic PW MPLS Control Word (CW), defined in [RFC4385], for DetNet displayed in Figure 1.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0|                Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: DetNet Control Word Format
PREOF in the DetNet domain composed by a combination of nodes that perform replication and elimination functions. The elimination function always uses the S-Label and packet sequencing information, e.g., the value in the Sequence Number field of DetNet CW (d-CW). The replication sub-function uses the S-Label information only. For data packets Figure 2 presents an example of PREOF in DetNet domain.

1111  1111111  1111111  112212  112212  132213
CE1----EN1--------R1-------R2-------R3--------EN2----CE2
\2          22222/                    \2
\222222  /----+                                  333333333333333333333333
+------R4------------------------+

Figure 2: DetNet Data Plane Based on PW

3.1. DetNet Active OAM Encapsulation

DetNet OAM, like PW OAM, uses PW Associated Channel Header defined in [RFC4385]. Figure 3 displays the encapsulation of a DetNet MPLS [RFC8964] active OAM packet.

+---------------------------------+
|                                 |
|        DetNet OAM Packet        |
|                                 |
|---------------------------------+ <--\ DetNet active OAM MPLS encapsulation
| DetNet Associated Channel Header|                  \[ F-Label(s) \]
| S-Label                         |
|---------------------------------+ <--/
| Data-Link                        |
|---------------------------------+                  \[ F-Label(s) \]
| Physical                           |
+---------------------------------+

Figure 3: DetNet Active OAM Packet Encapsulation in MPLS Data Plane

Figure 4 displays encapsulation of a test packet of an active DetNet OAM protocol in case of MPLS-over-UDP/IP [RFC9025].
Figure 4: DetNet Active OAM Packet Encapsulation in MPLS-over-UDP/IP

Figure 5 displays the format of the DetNet Associated Channel Header (d-ACH).

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 1|Version|Sequence Number|         Channel Type          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                 Node ID               |Level|  Flags  |Session|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: DetNet Associated Channel Header Format

The d-ACH encodes the following fields:

Bits 0..3 MUST be 0b0001. This value of the first nibble allows the packet to be distinguished from an IP packet [RFC4928] and a DetNet data packet [RFC8964].

Version - is a four-bits field, and the value is the version number of the d-ACH. This specification defines version 0x1.
Sequence Number - is an unsigned eight-bit field. The sequence number space is circular with no restriction on the initial value. The originator DetNet node MUST set the value of the Sequence Number field before the transmission of a packet. The originator node MUST increase the value of the Sequence Number field by 1 for each active OAM packet.

Channel Type - contains the value of DetNet Associated Channel Type. It is one of the values defined in the IANA PW Associated Channel Type registry.

Node ID - is an unsigned 20 bits-long field. The value of the Node ID field identifies the DetNet node that originated the packet. Methods of distributing Node ID are outside the scope of this specification.

Level - is a three-bits field.

Flags - is a five-bits field. Flags field contains five one-bit flags. Section 6.1 creates an IANA registry for new flags to be defined. Flags defined in this specification presented in Figure 6.

```
0 1 2 3 4
+-+-+-+-+-+
|U|U|U|U|U|
+-+-+-+-+-+
```

Figure 6: DetNet Associated Channel Header Flags Field Format

U: Unused and for future use. MUST be 0 on transmission and ignored on receipt.

Session ID is a four-bits field.

The DetNet flow, according to [RFC8964], is identified by the S-label that MUST be at the bottom of the stack. Active OAM packet MUST include d-ACH immediately following the S-label.
3.2. DetNet Packet Replication, Elimination, and Ordering Functions

Interaction with Active OAM

At the DetNet service sub-layer, special functions MAY be applied to the particular DetNet flow, PREOF, to potentially lower packet loss, improve the probability of on-time packet delivery and ensure in-order packet delivery. PREOF rely on sequencing information in the DetNet service sub-layer. For a DetNet active OAM packet, 28 MSBs of the d-ACH MUST be used as the source of the sequencing information by PREOF.

4. Use of Hybrid OAM in DetNet

Hybrid OAM methods are used in performance monitoring and defined in [RFC7799] as:

> Hybrid Methods are Methods of Measurement that use a combination of Active Methods and Passive Methods.

A hybrid measurement method may produce metrics as close to passive, but it still alters something in a data packet even if that is the value of a designated field in the packet encapsulation. One example of such a hybrid measurement method is the Alternate Marking method described in [RFC8321]. Reserving the field for the Alternate Marking method in the DetNet Header will enhance available to an operator set of DetNet OAM tools.

5. OAM Interworking Models

Interworking of two OAM domains that utilize different networking technology can be realized either by a peering or a tunneling model. In a peering model, OAM domains are within the corresponding network domain. When using the peering model, state changes that are detected by a Fault Management OAM protocol can be mapped from one OAM domain into another or a notification, e.g., an alarm, can be sent to a central controller. In the tunneling model of OAM interworking, usually, only one active OAM protocol is used. Its test packets are tunneled through another domain along with the data flow, thus ensuring the fate sharing among test and data packets.

5.1. OAM of DetNet MPLS Interworking with OAM of TSN

Active DetNet OAM is required to provide the E2E fault management and performance monitoring for a DetNet flow. Interworking of DetNet active OAM with MPLS data plane with the IEEE 802.1 Time-Sensitive Networking (TSN) domain based on [RFC9037].
In the case of the peering model is used in the fault management OAM, then the node that borders both TSN and DetNet MPLS domains MUST support [RFC7023]. [RFC7023] specified the mapping of defect states between Ethernet Attachment Circuits (ACs) and associated Ethernet PWs that are part of an end-to-end (E2E) emulated Ethernet service. Requirements and mechanisms described in [RFC7023] are equally applicable to using the peering model to achieve E2E FM OAM over DetNet MPLS and TSN domains. The Connectivity Fault Management (CFM) protocol [IEEE.CFM] or in [ITU.Y1731] can provide fast detection of a failure in the TSN segment of the DetNet service. In the DetNet MPLS domain BFD (Bidirectional Forwarding Detection), specified in [RFC5880] and [RFC5885], can be used. To provide E2E failure detection, the TSN segment might be presented as a concatenated with the DetNet MPLS and the Section 6.8.17 [RFC5880] MAY be used to inform the upstream DetNet MPLS node of a failure of the TSN segment. Performance monitoring can be supported by [RFC6374] in the DetNet MPLS and [ITU.Y1731] in the TSN domains, respectively. Performance objectives for each domain should refer to metrics that additive or be defined for each domain separately.

The following considerations are to be realized when using the tunneling model of OAM interworking between DetNet MPLS and TSN domains:

* Active OAM test packet MUST be mapped to the same TSN Stream ID as the monitored DetNet flow.

* Active OAM test packets MUST be treated in the TSN domain based on its S-label and CoS marking (TC field value).

Note that the tunneling model of the OAM interworking requires that the remote peer of the E2E OAM domain supports the active OAM protocol selected on the ingress endpoint. For example, if BFD is used for proactive path continuity monitoring in the DetNet MPLS domain, a TSN endpoint of the DetNet service has also support BFD as defined in [RFC5885].

5.2. OAM of DetNet MPLS Interworking with OAM of DetNet IP

Interworking between active OAM segments in DetNet MPLS and DetNet IP domains can also be realized using either the peering or the tunneling model, as discussed in Section 5.1. Using the same protocol, e.g., BFD, over both segments, simplifies the mapping of errors in the peering model. To provide the performance monitoring over a DetNet IP domain STAMP [RFC8762] and its extensions [RFC8972] can be used.
6. IANA Considerations

6.1. DetNet MPLS OAM Flags Registry

This document describes a new IANA-managed registry to identify DetNet MPLS OAM Flags Bits. The registration procedure is "IETF Review" [RFC8126]. The registry name is "DetNet MPLS OAM Flags". There are five flags in the five-bit Flags field, defined as in Table 1.

+-----+-------------+---------------+
| Bit | Description | Reference     |
+-----+-------------+---------------+
| 0-4 | Unassigned  | This document |
+-----+-------------+---------------+

Table 1: DetNet MPLS OAM Flags

7. Security Considerations

Additionally, security considerations discussed in DetNet specifications: [RFC8655], [RFC9055], [RFC8964] are applicable to this document. Security concerns and issues related to MPLS OAM tools like LSP Ping [RFC8029], BFD over PW [RFC5885] also apply to this specification.

8. Acknowledgment

Authors extend their appreciation to Pascal Thubert for his insightful comments and productive discussion that helped to improve the document.

9. References

9.1. Normative References


9.2. Informational References


Authors’ Addresses

Greg Mirsky
Ericsson
Email: gregimirsky@gmail.com

Mach(Guoyi) Chen
Huawei
Email: mach.chen@huawei.com

Balazs Varga
Ericsson
Budapest
Magyar Tudosok krt. 11.
1117
Hungary
Email: balazs.a.varga@ericsson.com

Janos Farkas
Ericsson
Budapest
Magyar Tudosok krt. 11.
1117
Hungary
Email: janos.farkas@ericsson.com
Asynchronous Deterministic Networking Framework for Large-Scale Networks
draft-joung-detnet-asynch-detnet-framework-00

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 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 December 28, 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
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 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 a 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. Therefore, 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, 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, and the port-based flow aggregate regulators (PFAR) [ADN] are proposed as solutions. The key component of the ATS and the FAIR frameworks is the interleaved regulator (IR). 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.

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.

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 consider 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, and therefore accumulation
over the flow’s path, which again contributes to the latency bound
increment. 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:

\[
B_{out} < B_{in} + (n-1)Lr/C,
\]

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,

\[
B_{out} < B_{in} + (n-1)B_{in}r/C.
\]
The second product term on the right-hand side represents the amount of increased maximum burst. It is dominated by the term \((n-1)B_{in}\), 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 regulation, or shaping, is the proposed key component in this document for such protection. It is the direct solution to decrease the accumulated burst into a desirable level. 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 HQQ, 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.

![FAIR Framework Diagram]

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:

\[
B_{out} < B_{in} + (p-1)B*r/C,
\]

where \( B_{in} \) 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:

\[
B_{out} < B_{in} + (p-1)B*r/C,
\]

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

\[
D < (B_{in} - B)/r,
\]

where \( D \) is the latency within the regulator.

Note that \( B_{out} \) is a function of \((n-1)B\), not \((n-1)B_{in}\); 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.

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) instances 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.
The buffer supports as many as the number of the flows destined for the destination. In cases where the buffer is not suitable to be placed within an end station, the network can attach a buffering function at the boundary. The destination shown in Figure 2 can also be a small deterministic network. There is an entity for time-stamping the arrival instances to the packets. The time-stamping function can use the real-time transport protocol (RTP) over the user datagram protocol 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 instance from the source, which is only a propagation time away from the packet arrival instance 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.

In the BN framework in Figure 2, it is recommended that the latency-lower bound information be provided by the network. The lower bound may be attributed to transmission and propagation delays within the network. The buffer holds packets in a flow according to the predefined intervals. The decision of the buffering intervals involves the timestamp within each packet.

5.3. Properties of the BN

Let the arrival instance 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 instances 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 instance 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 \( g \) includes the time to look up the timestamp and to store/forward the packet. It does not include an intentional buffer-holding interval. By definition, \( cn - bn \geq gn \). 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.

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

- \( c_1 = (b_1 + m - W) \),

- \( c_n = \max((bn + g), (c_1 + an - a_1)) \), for \( n > 1 \).

Furthermore, \( m \) is required to meet the inequality \( m \geq W + g \).

The second rule governing the \( c_n \) implies that a packet should be held in the buffer to make its inter-buffer-out time, \( (cn - c_1) \), equal to the inter-arrival time, \( (an - a_1) \). 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 \( a_n \) or \( a_1 \). 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 instances \( (cn - an) \), 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 instances \( (cn - an) \), 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 \( W \), then the jitter bound becomes \( (U + g - W) \), 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 tistemps 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 instances.

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.liu-detnet-large-scale-requirements]
Liu, P., Li, Y., Eckert, T., Xiong, Q., and J. Ryoo, "Requirements for Large-Scale Deterministic Networks", draft-liu-detnet-large-scale-requirements-02 (work in progress), April 2022.
10.2. Informative References


Li, Y., Ren, S., Li, G., Yang, F., Ryoo, J., and P. Liu, "IPv6 Options for Cyclic Queuing and Forwarding Variants", draft-yizhou-detnet-ipv6-options-for-cqf-variant-00 (work in progress), June 2022.


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
Deadline Based Deterministic Forwarding
draft-peng-detnet-deadline-based-forwarding-02

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 January 9, 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 Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ........................................... 2
   1.1. Requirements Language ............................ 3
2. Deadline Queue ......................................... 3
3. Get Deadline Information of Packets .................... 6
   3.1. Get Planned Residence Time ......................... 6
   3.2. Get Existing Cumulative Planned Residence Time ....... 7
   3.3. Get Existing Accumulated Actual Residence Time ....... 7
   3.4. Get Existing Accumulated Residence Time Deviation .... 8
4. Put Packets into the Deadline Queues .................. 8
5. Traffic Regulation and Shaping .......................... 11
6. Compatibility Considerations ............................ 12
7. Benefits .................................................. 14
8. IANA Considerations ..................................... 14
9. Security Considerations .................................. 15
10. Acknowledgements ....................................... 15
11. References ................................................ 15
   11.1. Normative References ............................... 15
   11.2. Informative References ............................. 15
Authors’ Addresses ........................................ 16

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, so a special data structure is recommended. 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 transmission 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 delay and avoids relying on time synchronization between nodes.

This document describes an alternate packets scheduling scheme that is 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 deadline time (also termed as TTL) are introduced and maintained for each outgoing port. These queues are called deadline queue and participate in priority based scheduling. Deadline queue has the following characteristics:

- The TTL 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 large 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, 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 TTL of all deadline queues, that is, whenever the timer expires, the deadline values of all queues will be subtracted from the timer interval. Note that the time interval of the timer must be less than or equal to the authorization time.

- For a deadline queue whose TTL has been reduced to 0, after a new round of authorization time, the TTL will return to the maximum initial value, 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 TTL 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 TTL value of the outgoing port for transmission.

- For a deadline queue whose TTL is not reduced to 0, its scheduling priority cannot be set to the highest value. The smaller the TTL, 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 TTL becomes 0.

In-time mode is applicable to low delay services, and on-time mode is applicable to low delay jitter services. Only one mode can be configured for a 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 TTL values, i.e., staggered from each other, so that the TTL of only one deadline queue will decrease to 0 at any time.

The above authorization time, timer interval and maximum initial TTL value shall be specified according to the actual capacity of the node. In fact, each node in the network can independently use different authorization time for different outgoing ports. The general principle is that if an outgoing port has a large bandwidth (such as 100G bps), the authorization time 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 authorization time should be larger (e.g. 10us), because the link with small bandwidth needs to send the required bit 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 authorization time is larger than the transmission time of a single packet, especially the authorization time should include the interference delay caused by a single low priority packet with maximum size.

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 authorization time 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 TTL attribute. The maximum initial TTL is 60us. At the initial time (T0), the TTL of all deadline queues are staggered from each other. For example, the TTL of queue-1 is 60us, the TTL of queue-2 is 50us, the TTL of queue-3 is 40us, and so on. At this time, only the TTL of queue-7 is 0, which has the highest scheduling priority.

Suppose the scheduling engine initiates a cycle timer with a time interval of 10us. After each timer timeout, the timer interval will be subtracted from the TTL of all deadline queues. As shown in the figure, at T0 + 10us, the timer timeout, the TTL of queue-1 becomes 50us, the TTL of queue-2 becomes 40us, the TTL of queue-3 becomes 30us, etc. At this time, the TTL of queue-7 returns to the maximum initial TTL of 60us and is no longer set to the highest scheduling priority; The TTL of queue-6 becomes 0, which has the highest scheduling priority.

When the TTL 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 TTL 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 for each deadline queue with specific TTL, if both in-time and on-time policies are supported, it 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 enters 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.

For a deterministic delay path based on deadline queue scheduling,
the path it passes through has deterministic end-to-end delay
requirements. It includes two parts, one is the cumulative node
delay and the other is the cumulative link transmission delay. The
end-to-end delay requirement is subtracted from the cumulative link
transmission delay to obtain the cumulative 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 Cumulative Planned Residence Time

The existing cumulative 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 this 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 cumulative planned residence time field in the packet.
[I-D.peng-6man-deadline-option] defined a method to carry existing
cumulative planned residence time in the IPv6 packets.

The setting of "existing cumulative 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 cumulative 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 this 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 cumulative actual residence time field in the packet. [I-D.peng-6man-deadline-option] defined a method to carry existing cumulative actual residence time in the IPv6 packets.

The setting of "existing cumulative 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 cumulative planned residence time minus existing cumulative actual residence time. This value can be positive or negative.

If the existing cumulative planned residence time and the existing cumulative 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.

4. Put Packets into the Deadline Queues

[I-D.ietf-detnet-bounded-latency] presents a latency model for DetNet nodes. There are six delays that a packet can experience from hop to hop. The processing delay (4), the regulator delay (5), the queuing subsystem delay (6), and the output delay (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 is obtained, and then the packet will be put to the deadline queue with TTL as the above allowable queuing delay value for sending. If the calculated allowable queuing delay is not exactly equal to the TTL of any queue, the packet selects the queue with the closest TTL to enter.

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, and the absolute value of the existing accumulated residence time deviation will be very small.

More generally, 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:

\[
E(i-1) = D(1) + D(2) + ... + D(i-1) - R(1) - R(2) - ... - R(i-1)
\]

\[
M(i) = D(i) + E(i-1)
\]

\[
Q(i) = M(i) - F(i)
\]

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 maximum initial TTL of the deadline queue maintained by the node, the allowable queuing delay (Q) should be modified to the maximum initial TTL.
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 TTL other than 0.

Figure 2 depicts an example of packets buffered to the 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 \( F = 10\text{us} \). 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 \(30 - 10 - 10 = 10\) us, and it will be put into the deadline queue-6 (its TTL is 10us).

- The allowable queuing delay (\(Q\)) of packet 2 in the node is \(20 + 10 - 10 = 20\) us, and it will be put into the deadline queue-5 (its TTL is 20us).

- The allowable queuing delay (\(Q\)) of packet 3 in the node is \(30 - 30 - 10 = -10\) us, and it will be modified to the minimum positive value 10 us then put into the deadline queue-6 (its TTL is 10us).

- The allowable queuing delay (\(Q\)) of packet 5 in the node is \(40 + 40 - 10 = 70\) us, and it will be modified to the maximum positive value 60 us then put into the deadline queue-1 (its TTL is 60us).

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

5. Traffic Regulation and Shaping

On the ingress PE node, traffic regulation is performed on UNI port, so that the service traffic does not exceed its reserved bandwidth. Suppose there are \(N\) sources, and the packets they send carry the same deadline. These packets may arrive at an intermediate node at the same time and put into the same deadline queue. If the reserved bandwidth of deadline queue at \(N\) sources is \(M_0\), and the reserved bandwidth of deadline queue at intermediate nodes is \(M_x\), then it needs to meet: \(N \cdot M_0 \leq M_x\). This means that a larger bandwidth is required on the intermediate node to send more bits in the same time duration, i.e., larger buffer size. 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.

On the ingress PE node, traffic shaping is performed on NNI 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 is equal to the reserved bandwidth * authorization time, however, at least one whole packet shall be loaded. For example, if the allowable queuing delay is 20us, then within the current authorization time, the first sequence of the packets will be put into the current deadline queue with TTL =
20us until the reserved bandwidth limit is reached; Then, within the next authorization time, the next sequence of packets will be put into the current TTL = 20us queue 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 shaping 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.

![Traffic Shaping Diagram]

Figure 3: Deadline Based Traffic Shaping

6. 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 4 shows an example of upgrading only several network edge 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. The delay analysis of in-time mode of deadline mechanism is similar to SP mechanism, except that in-time mode has different predefined upper latency bound (i.e., the planned residence time) and can further distinguish between emergency and non emergency according to deadline information other than traffic class. For example, a typical configuration is that if the burst interval is 250us and the planned residence time is 20us, each node within 12 hops will only face a single burst. In-time mode can even be pessimistic that its worst delay is the same as on-time mode, that is, the number of hops multiplied by the planned residence time. In later versions, we will further analyze how to ensure that the average actual residence time per hop does not exceed the planned residence time.

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

![Diagram](image-url)

Figure 4: Example of partial upgrade

7. 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 traverse 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.

8. IANA Considerations

There is no IANA request for this document.
9. Security Considerations

TBD

10. Acknowledgements

TBD

11. References

11.1. Normative References

[I-D.ietf-detnet-bounded-latency]

[I-D.peng-6man-deadline-option]

[I-D.stein-srtsn]


11.2. Informative References

[SP-LATENCY]
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
Abstract

This document introduces new IPv6 options to identify the DetNet queuing 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 11 January 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.
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. The [xiong-detnet-large-scale-enhancements] has proposed the overall framework of DetNet enhancements for large-scale deterministic networks. The packet treatment should support the queuing treatment and the identification of queuing 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 [IEEE802.1Qbv] and priority-scheduling includes the Credit-Based Shaper[IEEE802.1Q-2014] with Asynchronous Traffic Shaping[IEEE802.1Qcr]. The cyclic-scheduling queuing mechanism has

Table of Contents

1. Introduction .................................................. 2
2. Conventions used in this document .......................... 3
  2.1. Terminology ................................................. 3
  2.2. Requirements Language .................................... 3
3. The DetNet Options ............................................. 3
  3.1. The Queuing information Option ......................... 3
  3.1.1. Queuing Flag ............................................ 5
  3.1.2. Queuing Parameters ..................................... 6
4. Encapsulation of DetNet Options ............................. 7
  4.1. IPv6 Networks ................................................ 7
  4.2. SRv6 Networks .............................................. 7
5. Security Considerations ....................................... 8
6. Acknowledgements ............................................... 8
7. IANA Considerations ........................................... 8
  7.1. New Option for IPv6 ....................................... 8
8. Normative References .......................................... 9
Authors' Addresses ............................................. 10
been proposed in [IEEE802.1Qch] and improved in [I-D.dang-queuing-with-multiple-cyclic-buffers]. The deadline-scheduling queueing mechanism has been proposed in [I-D.stein-srtsn] and improved in [I-D.peng-detnet-deadline-based-forwarding]. The per-flow queueing mechanism includes Guaranteed-Service Integrated service (IntServ) [RFC2212]. It is required to carry queueing related information 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 follow the queue scheduling carried in the packet to achieve the end-to-end bounded latency.

This document introduces new IPv6 options to identify the DetNet queuing 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 queueing 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 Queuing information Option

The DetNet Queuing Information Option helps to discriminate the types of queuing mechanisms and specify the queuing parameters.
Option Type: 8-bit identifier of the type of option. Value TBD1 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 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.

Queuing Flag(16 bits): indicates the type of queuing mechanisms which can be used for the DetNet flows to identify the queuing information and the flag is defined in section 3.1.1.

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

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

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

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

Maximum Queuing Delay variation (variable): indicates the Maximum queuing delay variation. The value is various when the Queuing type is different.
Queuing Parameters Sub-TLVs (variable): it is optional and provides queuing information for a node to forward a DetNet flow. The Sub-TLVs has been defined in section 3.1.2.

3.1.1. Queuing Flag

The types of queuing mechanisms should cover time-scheduling, priority-scheduling, cyclic-scheduling, deadline-scheduling and per-flow scheduling queuing mechanisms. Other type of queuing mechanisms can be taken into considerations in the future. It indicates that a type of queuing mechanisms is used for DetNet when one flag is set to 1 and multiple 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.

The Flags field is designed as follow:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-----------------------------+
| T|P|C|A|M|D|                   |
+-----------------------------+
```

Figure 2: Queuing Flag

T flag: 1 bit, if the flag is set to 1, it indicates the TAS (Time Aware Shaping) [IEEE802.1Qbv] 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 such as queuing mechanisms defined in [I-D.peng-detnet-deadline-based-forwarding] and [I-D.stein-srtsn].
3.1.2. Queuing Parameters

The Queuing Parameters sub-TLVs is optional and it provides queuing information for a node to forward a DetNet flow. The format of Sub-TLVs is based on the queuing type and the flag field.

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

```
+-----------------------------------------------+  +-----------------------------------------------+
<p>| Sub-type | Length          | Sub-type | Length          |
|-----------------------------------------------|-----------------------------------------------|</p>
<table>
<thead>
<tr>
<th>Cycle Profile ID</th>
<th>Cycle Profile ID</th>
</tr>
</thead>
</table>
+-----------------------------------------------+  +-----------------------------------------------+
```

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:

```
+-----------------------------------------------+  +-----------------------------------------------+
| Sub-type | Length          | Sub-type | Length          |
|-----------------------------------------------|-----------------------------------------------|
| Deadline Information(variable)                | Deadline Information(variable)                |
+-----------------------------------------------+  +-----------------------------------------------+
```

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 Queuing 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.

```
+-----------------------------------+
<p>|         DetNet App-Flow           |</p>
<table>
<thead>
<tr>
<th>(original IP) Packet</th>
</tr>
</thead>
<tbody>
<tr>
<td>other EHs</td>
</tr>
</tbody>
</table>
+----------------------------------\|-----------------------------------+
| IPv6 Hop-by-Hop ExHdr            | DetNet Options |
| (DetNet Queuing information Option) |-----------------|
| IPv6 Header                      |
| Data-Link                        |
| Physical                          |
```

Figure 5: Queuing Information Option Format

4.2. SRv6 Networks

The DetNet Queuing 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 queuing information to achieve the packet forwarding and queue scheduling.
5. Security Considerations

TBA

6. Acknowledgements

The authors would like to thank Peng Liu, Bin Tan, Aihua Liu 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:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD1</td>
<td>Queuing Information Option</td>
<td>[this document]</td>
</tr>
</tbody>
</table>

Table 1

The Sub-TLVs has been proposed for Queuing Information Option as following shown:
Table 2

<table>
<thead>
<tr>
<th>Sub-type</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD2</td>
<td>Cycle Sub-TLV</td>
<td>[this document]</td>
</tr>
<tr>
<td>TBD3</td>
<td>Deadline Sub-TLV</td>
<td>[this document]</td>
</tr>
</tbody>
</table>

8. Normative References

[I-D.dang-queuing-with-multiple-cyclic-buffers]

[I-D.ietf-detnet-bounded-latency]

[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]

[I-D.peng-detnet-deadline-based-forwarding]
[I-D.stein-srtsn]


Authors’ Addresses

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

Xiong & Liu Expires 11 January 2023
DetNet Enhancements for Large-Scale Deterministic Networks
draft-xiong-detnet-large-scale-enhancements-00

Abstract

This document describes enhancements to DetNet to achieve the differentiated DetNet QoS in large-scale deterministic networks including the overall requirements and solutions with deterministic resources, routes and services.

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 5 August 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.

Xiong & Du Expires 28 August 2022 [Page 1]
1. Introduction

5G network is oriented to the internet of everything. In addition to the Enhanced Mobile Broadband (eMBB) and Massive Machine Type Communications (mMTC) services, it also supports the Ultra-reliable Low Latency Communications (uRLLC) services. The uRLLC services demand SLA guarantees such as low latency and high reliability and other deterministic and precise properties especially in Wide Area Network (WAN) applications.

The uRLLC services should be provided in large-scale networks which cover the industries such as intelligent electrical network, intelligent factory, internet of vehicles, industry automation and other industrial internet scenarios. The industrial internet is the key infrastructure that coordinate various units of work over various
system components, e.g. people, machines and things in the industrial environment including big data, cloud computing, Internet of Things (IOT), Augment Reality (AR), industrial robots, Artificial Intelligence (AI) and other basic technologies. For the intelligent electrical network, there are deterministic requirements for communication delay, jitter and packet loss rate. For example, in the electrical current difference model, a delay of 3~10ms and a jitter variation is no more than 100us are required. For the automation control, it is one of the basic application and the the core is closed-loop control system. The control process cycle is as low as millisecond level, so the system communication delay needs to reach millisecond level or even lower to ensure the realization of precise control. There are three levels of real-time requirements for industrial interconnection: factory level is about 1s, and process level is 10~100ms, and the highest real-time requirement is motion control, which requires less than 1ms.

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. The applications in 5G networks demand much more deterministic and precise properties in WAN. 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.

This document describes enhancements to DetNet to achieve the differentiated DetNet QoS in large-scale deterministic networks including the overall requirements and solutions with deterministic resources, routes and services.

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. DetNet Applicability for Large-Scale Deterministic Networks

As per [RFC8655], it defined the overall architecture for DetNet, which provides a capability for real-time applications with extremely low data loss rates and bounded latency within a network domain. It has three goals: minimum and maximum end-to-end latency from source to destination, bounded jitter (packet delay variation), packet loss ratio and upper bound on out-of-order packet delivery. To achieve the above objectives, multiple techniques need to be used in combination, including explicit routes, service protection and resource allocation defined by DetNet. And the DetNet functionality is implemented at DetNet service sub-layer and DetNet forwarding sub-layer. It is required to analyse the applicability in DetNet for large-scale deterministic networks.

From the perspective of services requirements discussed in section 4.1, a large-scale network needs to provide the deterministic service for various applications. And the deterministic service may demand different deterministic QoS requirements according to different application scenarios. The service protection in service sub-layer is not sufficient to meet the services requirements of large-scale networks, it should provide unified planning and scheduling mechanisms for service flows to perform end-to-end delay and jitter control and achieve differentiated DetNet QoS of multiple services.

The large-scale deterministic networks have a large number of hops and high link delay, which makes it difficult to achieve network-wide precise time synchronization. It may across multiple IP domains, or there may be different heterogeneous forwarding plane transport technologies. It is required to consider the efficiency of resources utilization and routes steering.

From the perspective of routes requirements discussed in section 4.3, a large-scale network should provide the deterministic paths for the services in large-scale networks. The deterministic routes should be calculated based on the deterministic metrics such as the end-to-end bounded latency and jitter. The forwarding sub-layer should establish the deterministic routes with SLA guarantees based on the deterministic resources. Moreover, other than explicit routes in centralized control scenarios, the distributed routes when the DetNet deployed with no controller may be more important for large-scale networks.

From the perspective of resources requirements discussed in section 4.3, a large-scale network should utilize the bandwidth, nodes, links, jitter resource, and queue scheduling resource and the other heterogeneous resources to establish the deterministic links which could provide SLA guarantees for the deterministic forwarding.
capabilities at different levels. Other than resource allocation, the forwarding sub-layer should support the unified and simplified scheduling and management mechanism for resources. For example, resource modeling, isolation and reservation should be considered to guarantee the deterministic transmission.

It is required to provide mechanisms within DetNet service and forwarding sub-layers to meet the requirements of large-scale deterministic networks. This document describes enhancements to DetNet to achieve the differentiated DetNet QoS in large-scale deterministic networks including the overall requirements and solutions with deterministic resources, routes and services.

4. Overall Requirements of Large-Scale Deterministic Networks

As per [draft-liu-detnet-large-scale-requirements], the technical and operational requirements have been specified for large-scale deterministic networks. For DetNet architecture to support deterministic service in a large-scale network, the requirements from services, routes and resources also need to be considered.

4.1. Service Requirements

4.1.1. Support the Differentiated DetNet QoS of Multiple Services

As defined in [RFC8655], the DetNet QoS can be expressed in terms of: Minimum and maximum end-to-end latency, bounded jitter (packet delay variation), packet loss ratio and an upper bound on out-of-order packet delivery. As described in [RFC8578], DetNet applications differ in their network topologies and specific desired behavior and different services requires differentiated DetNet QoS. In the large-scale networks, multiple services with differentiated DetNet QoS is co-existed in the same DetNet network. The classification of the deterministic flows within different levels is should be taken into considerations. It is required to provide Latency, bounded jitter and packet loss dynamically and flexibly in all scenarios for each characterized flow.

As the Figure 1 shows, the services can be divided into 5 levels and level 2~5 is the DetNet flows and level-1 is non-DetNet flow. DetNet applications and DetNet QoS is differentiated within each level.
From the perspective of deterministic service requirements, deterministic Quality of Service (QoS) in the network can be divided into five types or levels:

Level-1: bandwidth guarantee. The indicator requirements include basic bandwidth guarantee and certain packet loss tolerance. There is no requirement for the upper bound of the latency, and no requirement for the jitter. Typical services include download and FTP services.

Level-2: jitter guarantee. The indicator requirements include: jitter 50ms, delay 300ms. Typical services include synchronous voice services, such as voice call.

Level-3: Latency guarantee. The indicator requirements include: delay 50ms, jitter 50ms. Typical services include real-time communication services, such as video, production monitoring, and communication services.

Level-4: low delay and low jitter guarantee. The indicator requirements include: delay 20ms, jitter 5ms. Typical services include video interaction services, such as AR/VR, holographic communication, cloud video and cloud games.

Level-5: ultra-low delay and jitter guarantee. The indicator requirements include: delay 10ms, jitter 100us. Typical services include production control services, such as power protection and remote control.

Moreover, different DetNet services is required to tolerate different percentage of packet loss ratio such as 99.9%, 99.99%, 99.999%, and so on. It is also required to provide service isolation. In some scenarios, such as intelligent electrical network, the isolation...
requirements are very important. For example, the automatic operation or control of a process or isochronous data and service with different priorities need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services and hard isolated tunnel is required.

4.1.2. Guarantees of Multiple Dynamic Deterministic Flows

As described in [RFC8557], deterministic forwarding can only apply to flows with such well-defined characteristics as periodicity and burstiness. As defined in DetNet architecture [RFC8655], the traffic characteristics of an App-flow can be CBR (constant bit rate) or VBR (variable bit rate) of L1, L2 and L3 layers (VBR takes the maximum value when reserving resources). But the current scenarios and technical solutions only consider CBR flow, without considering the coexistence of VBR and CBR, the burst and aperiodicity of flows. The operations such as shaping or scheduling have not been specified. Even TSN mechanisms are based on a constant and forecastable traffic characteristics.

It will be more complicated in WAN applications where much more flows coexist and the traffic characteristics is more dynamic. A huge number of flows with different DetNet QoS requirements is dynamically concurrent and the state of each flow cannot be maintained. It is required to offer reliable delivery and SLA guarantee for dynamic flows. For example, periodic flow and aperiodic flow (including micro burst flow, etc.), CBR and VBR flow, flow with different periods or phases, etc. When the network needs to forward these deterministic flows at the same time, it must solve the problems of time micro bursts, queue processing and aggregation of multiple flows. It is required to guarantee the deterministic QoS of multiple dynamic flows. Flow shaping and concurrent and micro-burst control should be provided.

4.2. Route Requirements

Traditional routes only have reachability. Deterministic requirements such as delay and jitter are only used as path computation constraints. The paths vary with the real-time change of the network topology. They do not have Service Level Agreement (SLA) capability, and cannot meet the deterministic requirements at different levels. On the basic of the resources, the steering path and routes for deterministic flows should be programmed before the flows coming and able to provide SLA capability. And the routes should be considered to be established in distributed and centralized control Plane.
4.2.1. Support the Distributed Deterministic Routes

In large-scale deterministic networks, the distributed scenario with no controller should be taken into consideration. It is required to support the distributed deterministic routes which are established by distributed protocols such as IGP.

4.2.2. Support the Inter-domain Deterministic Routes

In large-scale deterministic networks, it may across multiple network domains, it is required to support the inter-domain deterministic routes to achieve the end-to-end latency, bounded jitter. And the deadline of latency and jitter of each domain and segment should be determined and controlled. The inter-domain mechanism MUST be considered at the boundary nodes such as BGP configurations.

4.2.3. Support the Replication and Elimination Deterministic Routes

As described in [RFC8557], the packet replication and elimination service protection should be provided to achieve the low packet loss ratio. It will copy the flows and spread the data over multiple disjoint forwarding paths. The bounded latency and jitter of each path should be meet service deterministic requirement. And the difference of latency within these paths should be limited. So the replication and elimination deterministic routes with configured latency and jitter policy should be supported.

4.3. Resource Requirements

4.3.1. Management and Scheduling of the Network Resources

Traditional Ethernet, IP and MPLS networks which is based on statistical multiplexing provides best-effort packet service and offers no delivery and SLA guarantee. As described in [RFC8655], the primary technique by which DetNet achieves its QoS is to allocate sufficient resources. But it can not be achieved by not sufficient resource which can be allocated due to practical and cost reason. So it is required to achieve the high-efficiency of resources utilization when provide the DetNet service.

Network resources include nodes, links, ports, bandwidth, queues, etc. The congestion control, shaping and queue scheduling and other traffic mechanisms which have been proposed in IEEE 802.1 TSN such as IEEE802.1Qbv, IEEE802.1Qch, IEEE802.1Qav, IEEE802.1Qcr and so on.

Resource classification and modeling is required along with the explicit path with more SLA guarantee parameters like bandwidth, latency, jitter, packet loss and so on.
4.3.2. Support the Utilization of Heterogeneous Resources

In large-scale application, a large-scale number of nodes and long-distance transmission in the network will lead to latency and jitter, such as increasing transmission latency, jitter and packet loss. It is required to reduce the scale of the network topology by establishing cut-through channels. The existing technologies such as FlexE and SR tunnels should be taken into consideration. And multiple capabilities is also provided by the nodes and links within the network topology such as FlexE tunnels, TSN sub-network and IP/MPLS/SRv6 tunnels. It is required to integrate the multi-capability resources to achieve the optimal DetNet QoS.

Heterogeneous resource should be used and unified and simplified resources mechanism under the selection of existing multiple technical methods to realize the elastic of deterministic capability.

5. Solutions of Large-Scale Deterministic Networks

5.1. Enhanced Layering Model

The large-scale IP network can provide three levels of determinism, deterministic resources, deterministic routes and deterministic services, to establish a unified large-scale deterministic IP network architecture. The deterministic resources maintains the resources of the entire network, and performs unified modeling for deterministic resources to form deterministic links to shield the differences in heterogeneous resource capabilities. The deterministic routes computes routes based on the deterministic links modeled at the resource layer to provide deterministic transport capabilities. The deterministic services performs traffic monitoring on ingress nodes by planning the traffic characteristics of service flows, and maps them to deterministic routes to meet the time requirements of different types and levels of services.
5.2. Mechanisms to Achieve Differentiated DetNet QoS

5.2.1. Deterministic Resources

Differentiated deterministic service requirements require the networks to provide different deterministic capabilities. The resources related to deterministic capabilities are also differentiated. The networks need to shield the differences between network capabilities. Deterministic resource is the basis for providing deterministic network services. It refers to the resources that meet the deterministic indicators of a node and link processing as well as the corresponding resource processing mechanisms (such as link bandwidth, queues, and scheduling algorithms). It is necessary to make overall resource planning for the network and make unified modeling for heterogeneous deterministic resources to form unified deterministic links to provide guarantee for the deterministic forwarding capabilities at different levels. A deterministic link can be a sub-network that provides deterministic transmission or a Point-to-Point (P2P) link. When the existing resources in the network are insufficient to meet the SLA requirements, virtual networks need to be reconstructed.
5.2.2. Deterministic Routes

To meet the requirements of different types and levels of deterministic services, deterministic route is to create deterministic routes with different SLA levels based on the deterministic link resources after unified modeling.

Deterministic routes can be based on strict explicit paths or loose routes. The former is applicable to centralized scenarios with controllers, and the latter is applicable to distributed scenarios without controllers. In the centralized scenario, when the source and sink PEs of a deterministic service are located at the two ends of a WAN with a limited physical range, one controller (single domain) or multiple controllers (cross domain) compute one or more paths with deterministic SLA in advance according to the typical Traffic Specification (T-SPEC) based on the collected deterministic resources, or compute dynamically according to the service T-SPEC as required by the services. It is suggested to generate two non-intersecting paths with very close delay to form 1+1 protection and perform concurrent transmission and dual reception, and make replication and elimination on the egress PE. In the distributed scenario, intrinsic deterministic loose routes are computed on the device side through routing protocols. Interior Gateway Protocol (IGP) is used to compute deterministic routes based on deterministic-delay inside a domain, and Border Gateway Protocol (BGP) is used to compute deterministic routes based on accurate delay/jitter across domains.

5.2.3. Deterministic Services

Deterministic services provide unified planning and scheduling mechanisms for service flows and perform end-to-end delay and jitter control. It is necessary to implement admission control and traffic policing at the ingress PE node based on the SLA of deterministic service flows, and map the service flows to deterministic routes to achieve the final goal of deterministic QoS.

Deterministic services support that the end-to-end delay/jitter of the traffic with a specific T-SPEC in the network will be strictly limited within a bounded range on the basis of deterministic resource and route. As different service levels have different requirements for delay and jitter, the resources and routing mechanisms used for mapping services to deterministic routes are also different. For example, the extremely low delay and jitter can be guaranteed by multiplexing the rigid pipes at L1, so as to avoid the excessive intra-node delay contributed by too many hops of intermediate nodes at L3. Or in the customized virtual network, the bounded delay and jitter can be guaranteed by forwarding along the paths composed of
links based on the ATS or CQF scheduling algorithm. Traffic policing on the ingress PE ensures that the service traffic does not exceed the reserved bandwidth, and then performs traffic shaping on the egress node. Different scheduling algorithms have different shaping effects.

6. Security Considerations

TBA

7. Acknowledgements

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

8. IANA Considerations

TBA

9. Normative References


Authors’ Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd

Xiong & Du Expires 28 August 2022 [Page 12]
Wuhan
Hubei, 430223
China
Email: xiong.quan@zte.com.cn

ZongPeng Du
China Mobile
Beijing
China
Email: duzongpeng@chinamobile.com
IPv6 Options for Cyclic Queuing and Forwarding Variants
draft-yizhou-detnet-ipv6-options-for-cqf-variant-00

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 December 22, 2022.
1. Introduction

The latency guarantee is 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 \((h-1) \times T_c + DT\) and \((h+1) \times T_c\) 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.

<table>
<thead>
<tr>
<th>Cycle Time (us)</th>
<th>Bytes Transmitted in a Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Link Speed</td>
</tr>
<tr>
<td></td>
<td>100Mbps</td>
</tr>
<tr>
<td>1</td>
<td>12.5</td>
</tr>
<tr>
<td>1.2</td>
<td>15</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
</tr>
<tr>
<td>4</td>
<td>50</td>
</tr>
<tr>
<td>10</td>
<td>125</td>
</tr>
<tr>
<td>12</td>
<td>150</td>
</tr>
<tr>
<td>120</td>
<td>1500*</td>
</tr>
</tbody>
</table>

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

DT=Dead Time
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 packets 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.
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.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|  Option Type  |  Opt Data Len |E|    Flags    |   Cycle Id    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
.                                                               .
--- (64-bit extension if flag E-bit is 1)                      .
.                                                               .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

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.
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.

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

8. 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 8.

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

Figure 8: CQF-Variant Option Code in Destination Options and Hop-by-Hop Options

9. Contributors

The following co-authors have contributed to this document:

Xiaoliang Zheng Huawei Email: zhengxiaoliang@huawei.com

10. References

10.1. Normative References


10.2. Informative References


[IEEE8021Q]

[multipleCQF]

[IPV6-PARMS]

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
DetNet Enhanced Data Plane
draft-yzz-detnet-enhanced-data-plane-00

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 12 January 2023.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
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 document 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.

<table>
<thead>
<tr>
<th>BLI Type</th>
<th>Format</th>
<th>Flag</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bounded Latency Information (variable size)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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,

<table>
<thead>
<tr>
<th>BLI Type Value</th>
<th>Bounded Latency Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>Time resource ID</td>
</tr>
<tr>
<td>2</td>
<td>Priority</td>
</tr>
<tr>
<td>3</td>
<td>End-to-end delay budget</td>
</tr>
<tr>
<td>4</td>
<td>Local delay budget</td>
</tr>
<tr>
<td>5</td>
<td>End-to-end deadline</td>
</tr>
<tr>
<td>6</td>
<td>Local deadline</td>
</tr>
<tr>
<td>7</td>
<td>End-to-end delay variation budget</td>
</tr>
<tr>
<td>8</td>
<td>Local delay variation budget</td>
</tr>
</tbody>
</table>
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,

<table>
<thead>
<tr>
<th>Format Value</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32-bit unsigned Integer</td>
</tr>
<tr>
<td>2</td>
<td>16-bit unsigned Integer</td>
</tr>
<tr>
<td>3</td>
<td>8-bit unsigned Integer</td>
</tr>
<tr>
<td>4</td>
<td>PTP 80-bit Timestamp</td>
</tr>
<tr>
<td>5</td>
<td>PTP 64-bit Timestamp</td>
</tr>
<tr>
<td>6</td>
<td>NTP 64-bit Timestamp</td>
</tr>
<tr>
<td>7</td>
<td>NTP 32-bit Timestamp</td>
</tr>
</tbody>
</table>

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 with Format Value 1.

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 with Format Value 3.

End-to-end delay budget: the end-to-end delay requirement of DetNet service packet. The format is 32-bit unsigned Integer with Format Value 1.
Local delay budget: the per hop delay requirement of DetNet service packet on this network node. The format is 32-bit unsigned Integer with Format Value 1.

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 8-bit or 16-bit unsigned Integer with Format Value 3 or Format Value 2.

Local delay variation budget: the per hop delay variation requirement of DetNet service packet on this network node. The format is 8-bit or 16-bit unsigned Integer with Format Value 3 or Format Value 2.

* 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.

```
0 1 2 3 4 5 6 7
+----------+
| U U U U U U U |
+----------+
```

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.

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
| Option Type       | Opt Data Len |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|   BLI Type    | Format    | Flag    | Reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
|                                                               |
| Bounded Latency Information (variable size)                   |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

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.

```
+---------------------------------------------------------------+
|                   NH          |      HLEN      |      EXT      |    Reserved   |
+---------------------------------------------------------------+
|   BLI Type    |     Format    |     Flag      |    Reserved   |
+---------------------------------------------------------------+
|                                                               |
+---------------------------------------------------------------+
|                                                            ~   |
|                                                            ~   |
+---------------------------------------------------------------+
```

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).
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:

<table>
<thead>
<tr>
<th>Suggested Value</th>
<th>Meaning</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Reserved</td>
<td>This I-D</td>
</tr>
<tr>
<td>TBD</td>
<td>Time resource ID</td>
<td>This I-D</td>
</tr>
<tr>
<td>TBD</td>
<td>Priority</td>
<td>This I-D</td>
</tr>
<tr>
<td>TBD</td>
<td>End-to-end delay budget</td>
<td>This I-D</td>
</tr>
<tr>
<td>TBD</td>
<td>Local delay budget</td>
<td>This I-D</td>
</tr>
<tr>
<td>TBD</td>
<td>End-to-end deadline</td>
<td>This I-D</td>
</tr>
<tr>
<td>TBD</td>
<td>Local deadline</td>
<td>This I-D</td>
</tr>
<tr>
<td>TBD</td>
<td>End-to-end delay variation budget</td>
<td>This I-D</td>
</tr>
<tr>
<td>TBD</td>
<td>Local delay variation budget</td>
<td>This I-D</td>
</tr>
</tbody>
</table>

### 7. Security Considerations

TBD

### 8. Normative References

[I-D.ietf-detnet-bounded-latency]

Yang, et al.   Expires 12 January 2023

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>.


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