

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 9, 2019

Z. Chen  
L. Qiang  
Huawei  
March 8, 2019

Deterministic VPN  
draft-chen-detnet-det-vpn-00

Abstract

Deterministic Networking (DetNet) services are expected to be integrated with VPN technologies. Such deployment scenarios include mobile backhauls and TSN islands inter-connections. This document describes an overall VPN framework that provides deterministic transport services (called deterministic VPN), specifies corresponding control and data plane protocols that are required to support the framework, and initially summarizes potential extensions of existing protocols to enable deterministic VPNs.

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 RFC 2119 [RFC2119].

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 September 9, 2019.

Copyright Notice

Copyright (c) 2019 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
2. Deployment Scenarios . . . . .	3
2.1. Mobile Backhaul . . . . .	3
2.2. Inter-connection of TSN Islands . . . . .	3
3. Deterministic VPN Framework . . . . .	4
3.1. Control Plane Protocols . . . . .	5
3.2. Data Plane Protocols . . . . .	6
4. IANA Considerations . . . . .	6
5. Security Considerations . . . . .	6
6. Acknowledgements . . . . .	6
7. Normative References . . . . .	6
Authors' Addresses . . . . .	7

## 1. Introduction

Deterministic Networking (DetNet) aims to provide bounded latency transport as well as low data loss rate for specific data flows over layer-3 networks [arch]. Potential use cases include electrical utilities, building automation systems, and industrial machine to machine [use-case]. From the perspective of real-world deployment, DetNet services are expected to be integrated with VPN technologies, e.g., MPLS/IP VPN, E-VPN. In particular, one or more VPN instances of an ISP network are required to provide bounded latency and low data loss rate transmission services for the customers. Such deployment scenarios include mobile backhauls and TSN islands inter-connections.

This document presents an overall VPN framework that provides deterministic transport services (called deterministic VPN), specifies corresponding control and data plane protocols that are required to support the framework, and initially summarizes potential extensions of existing protocols to enable deterministic VPNs. Note that specific extensions of existing protocols will be defined by separated documents.

## 2. Deployment Scenarios

This section introduces deployment scenarios for the deterministic VPN.

### 2.1. Mobile Backhaul

In the last decade, IP/MPLS devices are continuously replacing traditional TDM-based devices in operators' mobile backhaul networks for the benefits of simplicity and high bandwidth utilization. However, best effort IP forwarding cannot provide deterministic transport services as TDM could, making it hard to satisfy user's QoS requirements. DetNet is expected to fill up this gap. Simultaneously, layer-2 and layer-3 VPNs are also required in mobile backhaul networks in order to isolate different PDU sessions. Therefore, DetNet and VPN might be integrated in mobile backhaul networks.

The example in Figure 1 further illustrates such scenarios, where eNodeB and S-GW are connected by multiple IP routers (i.e., IP backhaul). In this case, a layer-3 or layer-2 deterministic VPN SHOULD be established between the eNodeB and the S-GW to carry the GTP-U encapsulation.

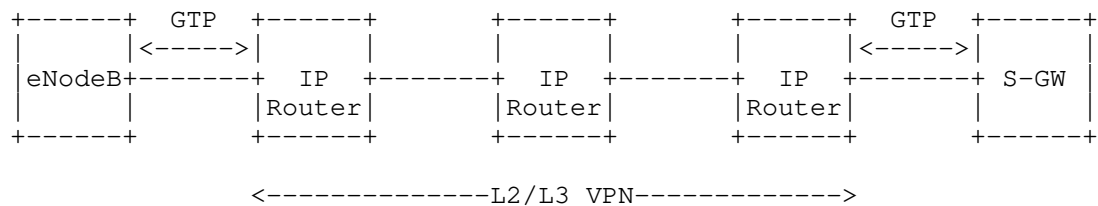


Figure 1

### 2.2. Inter-connection of TSN Islands

Another deployment scenario is using deterministic VPN to connect TSN islands. In particular, an enterprise may intent to connect its two (separately located) TSN networks by using an ISP network who can provide deterministic transport services. Since the ISP may provide the same service to different enterprises simultaneously, a layer-2 VPN SHOULD be established to isolate the traffic between different enterprises, as shown in Figure 2.

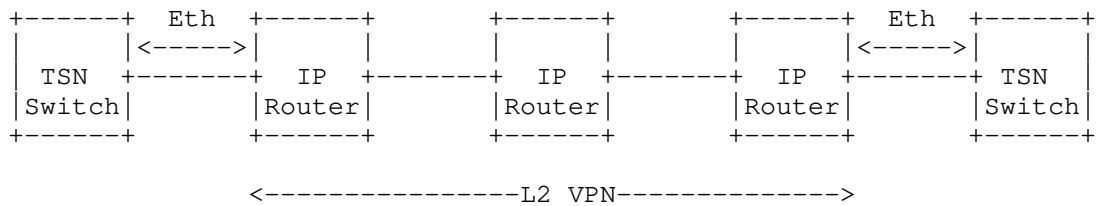


Figure 2

### 3. Deterministic VPN Framework

Figure 3 shows the overall framework of deterministic VPN, where CE1 and CE2 could be mobile network elements such as eNodeB, s-GW, or p-GW, or could be TSN switches of an enterprise network. PE1, P1, P2, and PE2 are ISP's IP/MPLS (layer-3) devices who have specific scheduling and shaping capabilities in the data plane thus providing deterministic transport (or DetNet) services. This document assumes that the CQF-based mechanism described in [ldn] is running on PE1, P1, P2 and PE2's data plane.

In this framework, a layer-2 or layer-3 VPN SHOULD be established between PE1 and PE2 to provide the deterministic transport service for CE1 and CE2. This requires that 1) data flows between CE1 and CE2 MUST be forwarded with bounded latency and low loss rate, and 2) the addresses as well as the traffic among CE1 and CE2 SHOULD be isolated from other CEs'. Note that although the overall framework is quite similar with existing VPN frameworks (e.g., IP/MPLS VPN and E-VPN), extensions of existing protocols are needed to support the deterministic transport, as will be discussed in the following subsections.

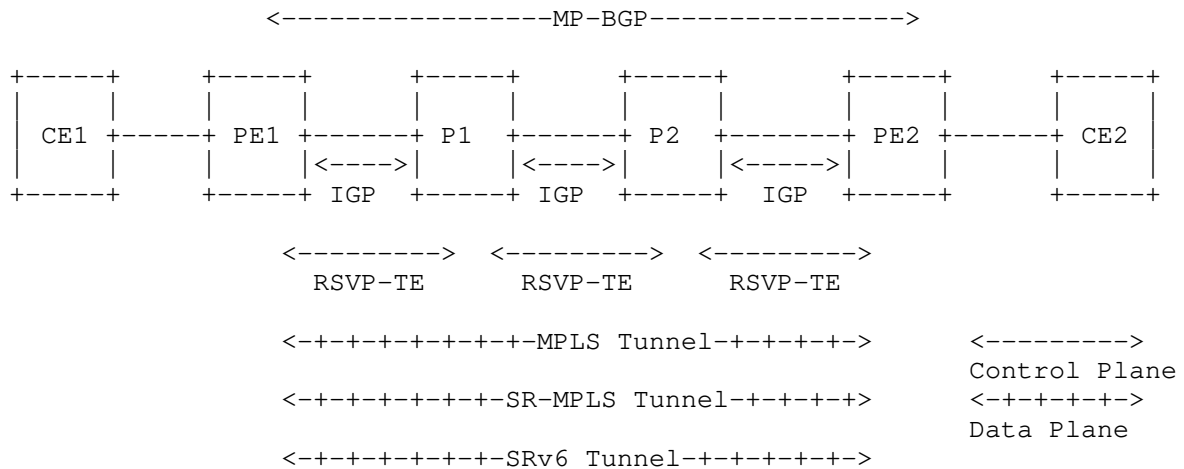


Figure 3

### 3.1. Control Plane Protocols

**IGP:** In the framework described above, IGP protocols such as IS-IS and OSPF SHOULD be used to advertise underlay reachability information. In the case where SR-MPLS and SRv6 encapsulations are chose for data plane tunnels, the IGP protocol SHOULD also advertise corresponding SIDs. To support deterministic VPN, corresponding information of deterministic transport, e.g., interface-level capability of scheduling and shaping, as well as available bandwidth SHOULD also be advertised by the IGP protocols [igp-te-ext].

**MP-BGP:** MP-BGP is used to advertise VPN reachability information in the framework, e.g., IP prefixes for layer-3 VPNs or MAC addresses for layer-2 VPNs, and corresponding VPN labels, e.g., MPLS labels or a SRv6 END.DX4 SIDs. To support deterministic VPN, MP-BGP SHOULD be extended to deliver related information for the deterministic transport services. Such extensions will be defined in separate documents.

**RSVP-TE:** RSVP-TE is used to reserve dedicated resources for the deterministic VPN flows. In the case where MPLS LSP is chose for the data plane encapsulation, RSVP-TE will also be used to allocate MPLS label(s) in each node along the forwarding path. To support deterministic VPN, RSVP-TE SHOULD be extended to carry relative information of the deterministic transport services. Such extensions will be defined in separate documents.

### 3.2. Data Plane Protocols

MPLS LSP Tunnel: If MPLS LSP tunnels are chose to be the data plane encapsulations for deterministic VPN flows, a mechanism of multiple MPLS labels per LSP per node SHOULD be used to identify different CQF forwarding cycles, as per [ldn]. Such mechanism has been described in [mpls-cqf].

SR-MPLS Tunnel: Accordingly, a SR-MPLS based mechanism to indicate different forwarding cycles at the data plane will also be specified in a separate document.

SRv6 Tunnel: Corresponding SRv6 based mechanism will also be specified in a separate document.

### 4. IANA Considerations

TBD.

### 5. Security Considerations

TBD.

### 6. Acknowledgements

TBD.

### 7. Normative References

- [arch] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", February 2019.
- [igp-te-ext] Geng, X., Chen, M., and Z. Li, "IGP-TE Extensions for DetNet Information Distribution", October 2018.
- [ldn] Qiang, L., Liu, B., Eckert, T., and L. Geng, "Large-Scale Deterministic Network", March 2019.
- [mpls-cqf] Chen, Z. and L. Qiang, "Multiple MPLS Labels for Cyclic Forwarding", March 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

[use-case]

Grossman, E., "Deterministic Networking Use Cases",  
December 2018.

Authors' Addresses

Zhe Chen  
Huawei

Email: [chenzhe17@huawei.com](mailto:chenzhe17@huawei.com)

Li Qiang  
Huawei

Email: [qiangli3@huawei.com](mailto:qiangli3@huawei.com)

DetNet  
Internet-Draft  
Intended status: Informational  
Expires: September 12, 2019

N. Finn  
Huawei Technologies Co. Ltd  
J-Y. Le Boudec  
E. Mohammadpour  
EPFL  
J. Zhang  
Huawei Technologies Co. Ltd  
B. Varga  
J. Farkas  
Ericsson  
March 11, 2019

DetNet Bounded Latency  
draft-finn-detnet-bounded-latency-03

Abstract

This document presents a parameterized timing model for Deterministic Networking (DetNet), so that existing and future standards can achieve the DetNet quality of service features of bounded latency and zero congestion loss. It defines requirements for resource reservation protocols or servers. It calls out queuing mechanisms, defined in other documents, that can provide the DetNet quality of service.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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 . . . . .	3
2. Terminology and Definitions . . . . .	4
3. DetNet bounded latency model . . . . .	4
3.1. Flow creation . . . . .	4
3.1.1. Static flow creation . . . . .	4
3.1.2. Dynamic flow creation . . . . .	5
3.2. Relay node model . . . . .	6
4. Computing End-to-end Latency Bounds . . . . .	9
4.1. Non-queuing delay bound . . . . .	9
4.2. Queuing delay bound . . . . .	9
4.2.1. Per-flow queuing mechanisms . . . . .	10
4.2.2. Per-class queuing mechanisms . . . . .	10
4.3. Ingress considerations . . . . .	11
4.4. Interspersed non-DetNet transit nodes . . . . .	11
5. Achieving zero congestion loss . . . . .	12
5.1. A General Formula . . . . .	12
6. Queuing model . . . . .	13
6.1. Queuing data model . . . . .	13
6.2. Preemption . . . . .	15
6.3. Time-scheduled queuing . . . . .	15
6.4. Time-Sensitive Networking with Asynchronous Traffic Shaping . . . . .	16
6.4.1. Flow Admission . . . . .	19
6.5. IntServ . . . . .	20
7. Time-based DetNet QoS . . . . .	22
7.1. Cyclic Queuing and Forwarding . . . . .	22
7.1.1. CQF timing sequence . . . . .	23
7.1.2. Dead time computation . . . . .	24
7.1.3. Tc computation . . . . .	24
7.1.4. CQF latency calculation . . . . .	24
7.1.5. CQF parameterization . . . . .	25
7.1.6. Ingress conditioning for CQF . . . . .	25
7.1.7. CQF ingress conditioning timing model . . . . .	27
7.2. Time-scheduled queuing . . . . .	28
8. Parameters for the bounded latency model . . . . .	29
9. References . . . . .	29

9.1. Normative References . . . . .	29
9.2. Informative References . . . . .	30
Authors' Addresses . . . . .	31

## 1. Introduction

The ability for IETF Deterministic Networking (DetNet) or IEEE 802.1 Time-Sensitive Networking (TSN, [IEEE8021TSN]) to provide the DetNet services of bounded latency and zero congestion loss depends upon A) configuring and allocating network resources for the exclusive use of DetNet/TSN flows; B) identifying, in the data plane, the resources to be utilized by any given packet, and C) the detailed behavior of those resources, especially transmission queue selection, so that latency bounds can be reliably assured. Thus, DetNet is an example of an IntServ Guaranteed Quality of Service [RFC2212]

As explained in [I-D.ietf-detnet-architecture], DetNet flows are characterized by 1) a maximum bandwidth, guaranteed either by the transmitter or by strict input metering; and 2) a requirement for a guaranteed worst-case end-to-end latency. That latency guarantee, in turn, provides the opportunity for the network to supply enough buffer space to guarantee zero congestion loss.

To be of use to the applications identified in [I-D.ietf-detnet-use-cases], it must be possible to calculate, before the transmission of a DetNet flow commences, both the worst-case end-to-end network latency, and the amount of buffer space required at each hop to ensure against congestion loss.

This document references specific queuing mechanisms, defined in other documents, that can be used to control packet transmission at each output port and achieve the DetNet qualities of service. This document presents a timing model for sources, destinations, and the DetNet transit nodes that relay packets that is applicable to all of those referenced queuing mechanisms. The parameters specified in this model:

- o Characterize a DetNet flow in a way that provides externally measurable verification that the sender is conforming to its promised maximum, can be implemented reasonably easily by a sending device, and does not require excessive over-allocation of resources by the network.
- o Enable reasonably accurate computation of worst-case end-to-end latency, in a way that requires as little detailed knowledge as possible of the behavior of the Quality of Service (QoS) algorithms implemented in each device, including queuing, shaping, metering, policing, and transmission selection techniques.

Using the model presented in this document, it should be possible for an implementor, user, or standards development organization to select a particular set of queuing mechanisms for each device in a DetNet network, and to select a resource reservation algorithm for that network, so that those elements can work together to provide the DetNet service.

This document does not specify any resource reservation protocol or server. It does not describe all of the requirements for that protocol or server. It does describe requirements for such resource reservation methods, and for queuing mechanisms that, if met, will enable them to work together.

## 2. Terminology and Definitions

This document uses the terms defined in [I-D.ietf-detnet-architecture].

## 3. DetNet bounded latency model

### 3.1. Flow creation

There are two models for flow creation, static (Section 3.1.1) and dynamic (Section 3.1.2). Most of the mathematical analysis provided in this document is applicable to either flow creation model; any dependencies on the choice of flow creation model are pointed out in the text.

#### 3.1.1. Static flow creation

The static problem:

Given a network and a set of DetNet flows, compute an end-to-end latency bound (if computable) for each flow, and compute the resources, particularly buffer space, required in each DetNet transit node to achieve zero congestion loss.

In this model, all of the DetNet flows are known before the calculation commences. This problem is of interest to relatively static networks, or static parts of larger networks. It gives the best possible worst-case behavior. The calculations can be extended to provide global optimizations, such as altering the path of one DetNet flow in order to make resources available to another DetNet flow with tighter constraints.

This calculation may be more difficult to perform than that of the dynamic model (Section 3.1.2), because the flows passing through one port on a DetNet transit node affect each others' latency. The effects can even be circular, from Flow A to B to C and back to A.

On the other hand, the static calculation can often accommodate queuing methods, such as transmission selection by strict priority, that are unsuitable for the dynamic calculation.

The static flow creation model is not limited only to static networks; the entire calculation for all flows can be repeated each time a new DetNet flow is created or deleted. If some already-established flow would be pushed beyond its latency requirements by the new flow, then either the new flow is refused, or some other suitable action taken.

### 3.1.2. Dynamic flow creation

The dynamic problem:

Given a network whose maximum capacity for DetNet flows is bounded by a set of static configuration parameters applied to the DetNet transit nodes, and given just one DetNet flow, compute the worst-case end-to-end latency that can be experienced by that flow, no matter what other DetNet flows (within the network's configured parameters) might be created or deleted in the future. Also, compute the resources, particularly buffer space, required in each DetNet transit node to achieve zero congestion loss.

This model is dynamic, in the sense that flows can be added or deleted at any time, with a minimum of computation effort, and without affecting the guarantees already given to other flows.

The choice of queuing methods is critical to the applicability of the dynamic model. Some queuing methods (e.g. CQF, Section 7.1) make it easy to configure bounds on the network's capacity, and to make independent calculations for each flow. Other queuing methods (e.g., transmission selection by strict priority), make this calculation impossible, because the worst case for one flow cannot be computed without complete knowledge of all other flows. Other queuing methods (e.g. the credit-based shaper defined in [IEEE8021Q] section 8.6.8.2) can be used for dynamic flow creation, but yield poorer latency and buffer space guarantees than when that same queuing method is used for static flow creation (Section 3.1.1).

The dynamic flow creation model assumes the use of the following paradigm for provisioning DetNet flows:

1. Perform any configuration required by the DetNet transit nodes in the network for the classes of service to be offered, including one or more classes of DetNet service. This configuration is done beforehand, and not tied to any particular flow.

2. Characterize the new DetNet flow in IntServ terms (Section 8).
3. Establish the path that the DetNet flow will take through the network from the source to the destination(s). This can be a point-to-point or a point-to-multipoint path.
4. Select one of the DetNet classes of service for the DetNet flow.
5. Compute the worst-case end-to-end latency for the DetNet flow. In the process, determine whether sufficient resources are available for that flow to guarantee the required latency and to provide zero congestion loss.
6. Assuming that the resources are available, commit those resources to the flow. This may or may not require adjusting the parameters that control the queuing mechanisms at each hop along the flow's path.

This paradigm can be static and/or dynamic, and can be implemented using peer-to-peer protocols or using a central server model. In some situations, backtracking and recursing through this list may be necessary.

Issues such as un-provisioning a DetNet flow in favor of another when resources are scarce are not considered, but are left to the static flow creation model (Section 3.1.1). How the path to be taken by a DetNet flow is chosen is not considered in this document.

### 3.2. Relay node model

A model for the operation of a DetNet transit node is required, in order to define the latency and buffer calculations. In Figure 1 we see a breakdown of the per-hop latency experienced by a packet passing through a DetNet transit node, in terms that are suitable for computing both hop-by-hop latency and per-hop buffer requirements.

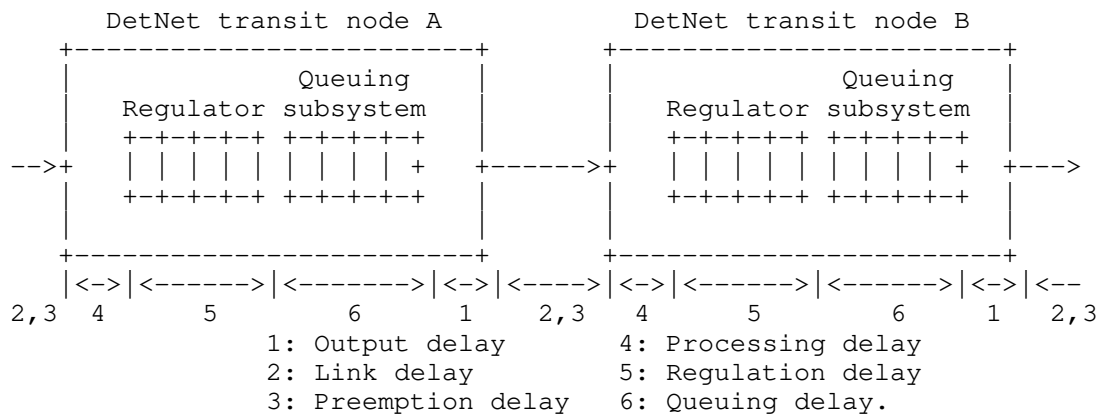


Figure 1: Timing model for DetNet or TSN

In Figure 1, we see two DetNet transit nodes (typically, bridges or routers), with a wired link between them. In this model, the only queues we deal with explicitly are attached to the output port; other queues are modeled as variations in the other delay times. (E.g., an input queue could be modeled as either a variation in the link delay [2] or the processing delay [4].) There are six delays that a packet can experience from hop to hop.

#### 1. Output delay

The time taken from the selection of a packet for output from a queue to the transmission of the first bit of the packet on the physical link. If the queue is directly attached to the physical port, output delay can be a constant. But, in many implementations, the queuing mechanism in a forwarding ASIC is separated from a multi-port MAC/PHY, in a second ASIC, by a multiplexed connection. This causes variations in the output delay that are hard for the forwarding node to predict or control.

#### 2. Link delay

The time taken from the transmission of the first bit of the packet to the reception of the last bit, assuming that the transmission is not suspended by a preemption event. This delay has two components, the first-bit-out to first-bit-in delay and the first-bit-in to last-bit-in delay that varies with packet size. The former is typically measured by the Precision Time Protocol and is constant (see [I-D.ietf-detnet-architecture]). However, a virtual "link" could exhibit a variable link delay.

#### 3. Preemption delay

If the packet is interrupted in order to transmit another packet or packets, (e.g. [IEEE8023] clause 99 frame preemption) an arbitrary delay can result.

4. Processing delay

This delay covers the time from the reception of the last bit of the packet to the time the packet is enqueued in the regulator (Queuing subsystem, if there is no regulation). This delay can be variable, and depends on the details of the operation of the forwarding node.

5. Regulator delay

This is the time spent from the insertion of the last bit of a packet into a regulation queue until the time the packet is declared eligible according to its regulation constraints. We assume that this time can be calculated based on the details of regulation policy. If there is no regulation, this time is zero.

6. Queuing subsystem delay

This is the time spent for a packet from being declared eligible until being selected for output on the next link. We assume that this time is calculable based on the details of the queuing mechanism. If there is no regulation, this time is from the insertion of the packet into a queue until it is selected for output on the next link.

Not shown in Figure 1 are the other output queues that we presume are also attached to that same output port as the queue shown, and against which this shown queue competes for transmission opportunities.

The initial and final measurement point in this analysis (that is, the definition of a "hop") is the point at which a packet is selected for output. In general, any queue selection method that is suitable for use in a DetNet network includes a detailed specification as to exactly when packets are selected for transmission. Any variations in any of the delay times 1-4 result in a need for additional buffers in the queue. If all delays 1-4 are constant, then any variation in the time at which packets are inserted into a queue depends entirely on the timing of packet selection in the previous node. If the delays 1-4 are not constant, then additional buffers are required in the queue to absorb these variations. Thus:

- o Variations in output delay (1) require buffers to absorb that variation in the next hop, so the output delay variations of the previous hop (on each input port) must be known in order to calculate the buffer space required on this hop.

- o Variations in processing delay (4) require additional output buffers in the queues of that same DetNet transit node. Depending on the details of the queueing subsystem delay (6) calculations, these variations need not be visible outside the DetNet transit node.

#### 4. Computing End-to-end Latency Bounds

##### 4.1. Non-queueing delay bound

End-to-end latency bounds can be computed using the delay model in Section 3.2. Here it is important to be aware that for several queueing mechanisms, the worst-case end-to-end delay is less than the sum of the per-hop worst-case delays. An end-to-end latency bound for one DetNet flow can be computed as

$$\text{end\_to\_end\_latency\_bound} = \text{non\_queueing\_latency} + \text{queueing\_latency}$$

The two terms in the above formula are computed as follows. First, at the  $h$ -th hop along the path of this DetNet flow, obtain an upper bound `per-hop_non_queueing_latency[h]` on the sum of delays 1,2,3,4 of Figure 1. These upper-bounds are expected to depend on the specific technology of the DetNet transit node at the  $h$ -th hop but not on the T-SPEC of this DetNet flow. Then set `non_queueing_latency` = the sum of `per-hop_non_queueing_latency[h]` over all hops  $h$ .

##### 4.2. Queueing delay bound

Second, compute `queueing_latency` as an upper bound to the sum of the queueing delays along the path. The value of `queueing_latency` depends on the T-SPEC of this flow and possibly of other flows in the network, as well as the specifics of the queueing mechanisms deployed along the path of this flow.

For several queueing mechanisms, `queueing_latency` is less than the sum of upper bounds on the queueing delays (5,6) at every hop. This occurs with (1) per-flow queueing, and (2) per-class queueing with regulators, as explained in Section 4.2.1, Section 4.2.2, and Section 6.

For other queueing mechanisms the only available value of `queueing_latency` is the sum of the per-hop queueing delay bounds. In such cases, the computation of per-hop queueing delay bounds must account for the fact that the T-SPEC of a DetNet flow is no longer satisfied at the ingress of a hop, since burstiness increases as one flow traverses one DetNet transit node.



#### 4.2.1. Per-flow queuing mechanisms

With such mechanisms, each flow uses a separate queue inside every node. The service for each queue is abstracted with a guaranteed rate and a delay. For every flow the per-node delay bound as well as end-to-end delay bound can be computed from the traffic specification of this flow at its source and from the values of rates and latencies at all nodes along its path. Details of calculation for IntServ are described in Section 6.5.

#### 4.2.2. Per-class queuing mechanisms

With such mechanisms, the flows that have the same class share the same queue. A practical example is the queuing mechanism in Time Sensitive Networking. One key issue in this context is how to deal with the burstiness cascade: individual flows that share a resource dedicated to a class may see their burstiness increase, which may in turn cause increased burstiness to other flows downstream of this resource. Computing latency upper bounds for such cases is difficult, and in some conditions impossible [charny2000delay][bennett2002delay]. Also, when bounds are obtained, they depend on the complete configuration, and must be recomputed when one flow is added.

A solution to deal with this issue is to reshape the flows at every hop. This can be done with per-flow regulators (e.g. leaky bucket shapers), but this requires per-flow queuing and defeats the purpose of per-class queuing. An alternative is the interleaved regulator, which reshapes individual flows without per-flow queuing ([Specht2016UBS], [IEEE8021Qcr]). With an interleaved regulator, the packet at the head of the queue is regulated based on its (flow) regulation constraints; it is released at the earliest time at which this is possible without violating the constraint. One key feature of per-flow or interleaved regulator is that, it does not increase worst-case latency bounds [le\_boudec\_theory\_2018]. Specifically, when an interleaved regulator is appended to a FIFO subsystem, it does not increase the worst-case delay of the latter.

Figure 2 shows an example of a network with 5 nodes, per-class queuing mechanism and interleaved regulators as in Figure 1. An end-to-end delay bound for flow  $f$ , traversing nodes 1 to 5, is calculated as follows:

$$\text{end\_to\_end\_latency\_bound\_of\_flow\_f} = C_{12} + C_{23} + C_{34} + S_4$$

In the above formula,  $C_{ij}$  is a bound on the aggregate response time of queuing subsystem in node  $i$  and interleaved regulator of node  $j$ , and  $S_4$  is a bound on the response time of the queuing subsystem in

node 4 for flow  $f$ . In fact, using the delay definitions in Section 3.2,  $C_{ij}$  is a bound on sum of the delays 1,2,3,6 of node  $i$  and 4,5 of node  $j$ . Similarly,  $S_4$  is a bound on sum of the delays 1,2,3,6 of node 4. A practical example of queuing model and delay calculation is presented Section 6.4.

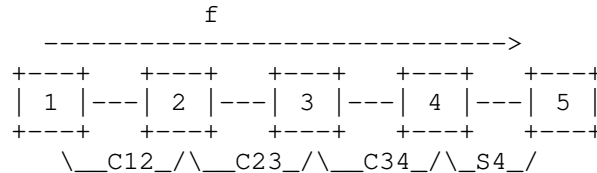


Figure 2: End-to-end latency computation example

REMARK: The end-to-end delay bound calculation provided here gives a much better upper bound in comparison with end-to-end delay bound computation by adding the delay bounds of each node in the path of a flow [TSNwithATS].

#### 4.3. Ingress considerations

A sender can be a DetNet node which uses exactly the same queuing methods as its adjacent DetNet transit node, so that the latency and buffer calculations at the first hop are indistinguishable from those at a later hop within the DetNet domain. On the other hand, the sender may be DetNet unaware, in which case some conditioning of the flow may be necessary at the ingress DetNet transit node.

This ingress conditioning typically consists of a FIFO with an output regulator that is compatible with the queuing employed by the DetNet transit node on its output port(s). For some queuing methods, simply requires added extra buffer space in the queuing subsystem. Ingress conditioning requirements for different queuing methods are mentioned in the sections, below, describing those queuing methods.

#### 4.4. Interspersed non-DetNet transit nodes

It is sometimes desirable to build a network that has both DetNet aware transit nodes and DetNet non-aware transit nodes, and for a DetNet flow to traverse an island of non-DetNet transit nodes, while still allowing the network to offer latency and congestion loss guarantees. This is possible under certain conditions.

In general, when passing through a non-DetNet island, the island causes delay variation in excess of what would be caused by DetNet nodes. That is, the DetNet flow is "lumpier" after traversing the non-DetNet island. DetNet guarantees for latency and buffer

requirements can still be calculated and met if and only if the following are true:

1. The latency variation across the non-DetNet island must be bounded and calculable.
2. An ingress conditioning function (Section 4.3) may be required at the re-entry to the DetNet-aware domain. This will, at least, require some extra buffering to accommodate the additional delay variation, and thus further increases the worst-case latency.

The ingress conditioning is exactly the same problem as that of a sender at the edge of the DetNet domain. The requirement for bounds on the latency variation across the non-DetNet island is typically the most difficult to achieve. Without such a bound, it is obvious that DetNet cannot deliver its guarantees, so a non-DetNet island that cannot offer bounded latency variation cannot be used to carry a DetNet flow.

## 5. Achieving zero congestion loss

When the input rate to an output queue exceeds the output rate for a sufficient length of time, the queue must overflow. This is congestion loss, and this is what deterministic networking seeks to avoid.

### 5.1. A General Formula

To avoid congestion losses, an upper bound on the backlog present in the regulator and queuing subsystem of Figure 1 must be computed during resource reservation. This bound depends on the set of flows that use these queues, the details of the specific queuing mechanism and an upper bound on the processing delay (4). The queue must contain the packet in transmission plus all other packets that are waiting to be selected for output.

A conservative backlog bound, that applies to all systems, can be derived as follows.

The backlog bound is counted in data units (bytes, or words of multiple bytes) that are relevant for buffer allocation. For every class we need one buffer space for the packet in transmission, plus space for the packets that are waiting to be selected for output. Excluding transmission and preemption times, the packets are waiting in the queue since reception of the last bit, for a duration equal to the processing delay (4) plus the queuing delays (5,6).

Let

- o `nb_classes` be the number of classes of traffic that may use this output port
- o `total_in_rate` be the sum of the line rates of all input ports that send traffic of any class to this output port. The value of `total_in_rate` is in data units (e.g. bytes) per second.
- o `nb_input_ports` be the number input ports that send traffic of any class to this output port
- o `max_packet_length` be the maximum packet size for packets of any class that may be sent to this output port. This is counted in data units.
- o `max_delay45` be an upper bound, in seconds, on the sum of the processing delay (4) and the queuing delays (5,6) for a packet of any class at this output port.

Then a bound on the backlog of traffic of all classes in the queue at this output port is

$$\text{backlog\_bound} = ( \text{nb\_classes} + \text{nb\_input\_ports} ) * \text{max\_packet\_length} + \text{total\_in\_rate} * \text{max\_delay45}$$

## 6. Queuing model

### 6.1. Queuing data model

Sophisticated queuing mechanisms are available in Layer 3 (L3, see, e.g., [RFC7806] for an overview). In general, we assume that "Layer 3" queues, shapers, meters, etc., are precisely the "regulators" shown in Figure 1. The "queuing subsystems" in this figure are not the province solely of bridges; they are an essential part of any DetNet transit node. As illustrated by numerous implementation examples, some of the "Layer 3" mechanisms described in documents such as [RFC7806] are often integrated, in an implementation, with the "Layer 2" mechanisms also implemented in the same node. An integrated model is needed in order to successfully predict the interactions among the different queuing mechanisms needed in a network carrying both DetNet flows and non-DetNet flows.

Figure 3 shows the general model for the flow of packets through the queues of a DetNet transit node. Packets are assigned to a class of service. The classes of service are mapped to some number of regulator queues. Only DetNet/TSN packets pass through regulators. Queues compete for the selection of packets to be passed to queues in the queuing subsystem. Packets again are selected for output from the queuing subsystem.

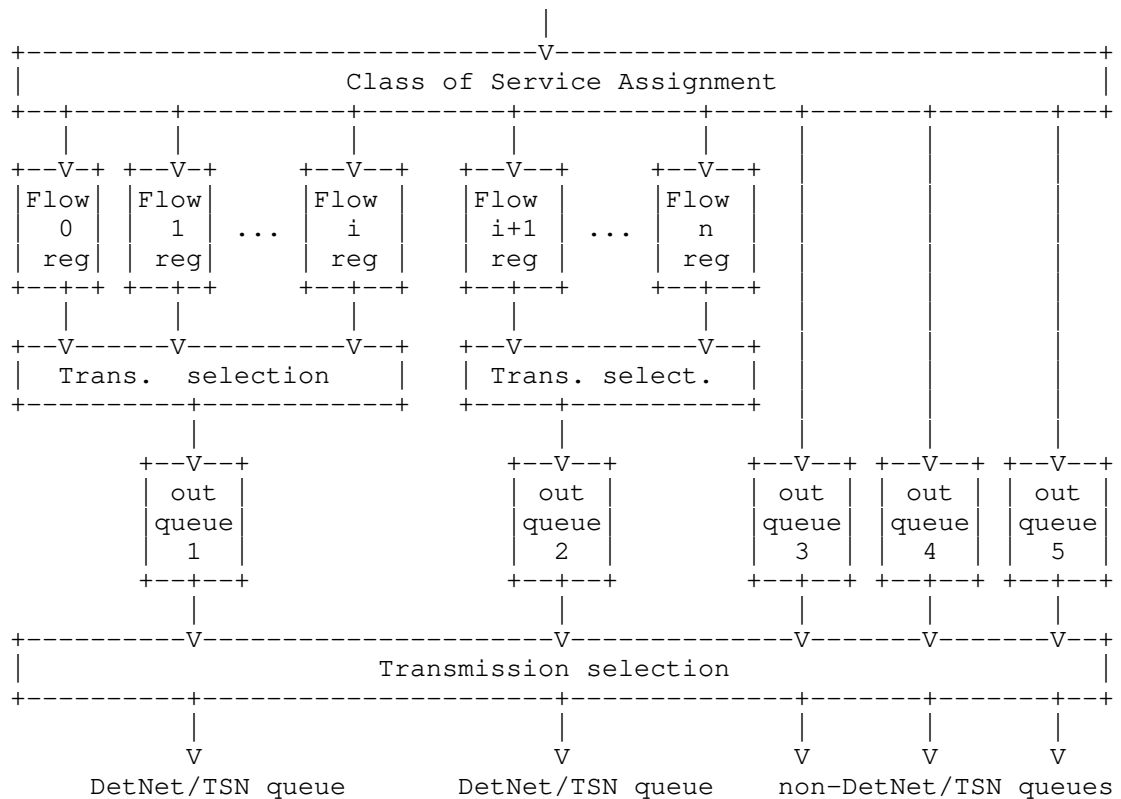


Figure 3: IEEE 802.1Q Queuing Model: Data flow

Some relevant mechanisms are hidden in this figure, and are performed in the queue boxes:

- o Discarding packets because a queue is full.
- o Discarding packets marked "yellow" by a metering function, in preference to discarding "green" packets.

Ideally, neither of these actions are performed on DetNet packets. Full queues for DetNet packets should occur only when a flow is misbehaving, and the DetNet QoS does not include "yellow" service for packets in excess of committed rate.

The Class of Service Assignment function can be quite complex, even in a bridge [IEEE8021Q], since the introduction of per-stream filtering and policing ([IEEE8021Q] clause 8.6.5.1). In addition to the Layer 2 priority expressed in the 802.1Q VLAN tag, a DetNet

transit node can utilize any of the following information to assign a packet to a particular class of service (queue):

- o Input port.
- o Selector based on a rotating schedule that starts at regular, time-synchronized intervals and has nanosecond precision.
- o MAC addresses, VLAN ID, IP addresses, Layer 4 port numbers, DSCP. ([I-D.ietf-detnet-dp-sol-ip], [I-D.ietf-detnet-dp-sol-mpls]) (Work items are expected to add MPC and other indicators.)
- o The Class of Service Assignment function can contain metering and policing functions.
- o MPLS and/or pseudowire ([RFC6658]) labels.

The "Transmission selection" function decides which queue is to transfer its oldest packet to the output port when a transmission opportunity arises.

## 6.2. Preemption

In [IEEE8021Q] and [IEEE8023], the transmission of a frame can be interrupted by one or more "express" frames, and then the interrupted frame can continue transmission. This frame preemption is modeled as consisting of two MAC/PHY stacks, one for packets that can be interrupted, and one for packets that can interrupt the interruptible packets. The Class of Service (queue) determines which packets are which. Only one layer of preemption is supported -- a transmitter cannot have more than one interrupted frame in progress. DetNet flows typically pass through the interrupting MAC. Best-effort queues pass through the interruptible MAC, and can thus be preempted.

## 6.3. Time-scheduled queuing

In [IEEE8021Q], the notion of time-scheduling queue gates is described in section 8.6.8.4. Below every output queue (the lower row of queues in Figure 3) is a gate that permits or denies the queue to present data for transmission selection. The gates are controlled by a rotating schedule that can be locked to a clock that is synchronized with other DetNet transit nodes. The DetNet class of service can be supplied by queuing mechanisms based on time, rather than the regulator model in Figure 3. These queuing mechanisms are discussed in Section 7, below.

#### 6.4. Time-Sensitive Networking with Asynchronous Traffic Shaping

Consider a network with a set of nodes (DetNet transit nodes and hosts) along with a set of flows between hosts. Hosts are sources or destinations of flows. There are four types of flows, namely, control-data traffic (CDT), class A, class B, and best effort (BE) in decreasing order of priority. Flows of classes A and B are together referred to AVB flows. It is assumed a subset of TSN functions as described next.

It is also assumed that contention occurs only at the output port of a TSN node. Each node output port performs per-class scheduling with eight classes: one for CDT, one for class A traffic, one for class B traffic, and five for BE traffic denoted as BE0-BE4 (according to TSN standard). In addition, each node output port also performs per-flow regulation for AVB flows using an interleaved regulator (IR), called Asynchronous Traffic Shaper (ATS) in TSN. Thus, at each output port of a node, there is one interleaved regulator per-input port and per-class. The detailed picture of scheduling and regulation architecture at a node output port is given by Figure 4. The packets received at a node input port for a given class are enqueued in the respective interleaved regulator at the output port. Then, the packets from all the flows, including CDT and BE flows, are enqueued in a class based FIFO system (CBFS) [TSNwithATS].

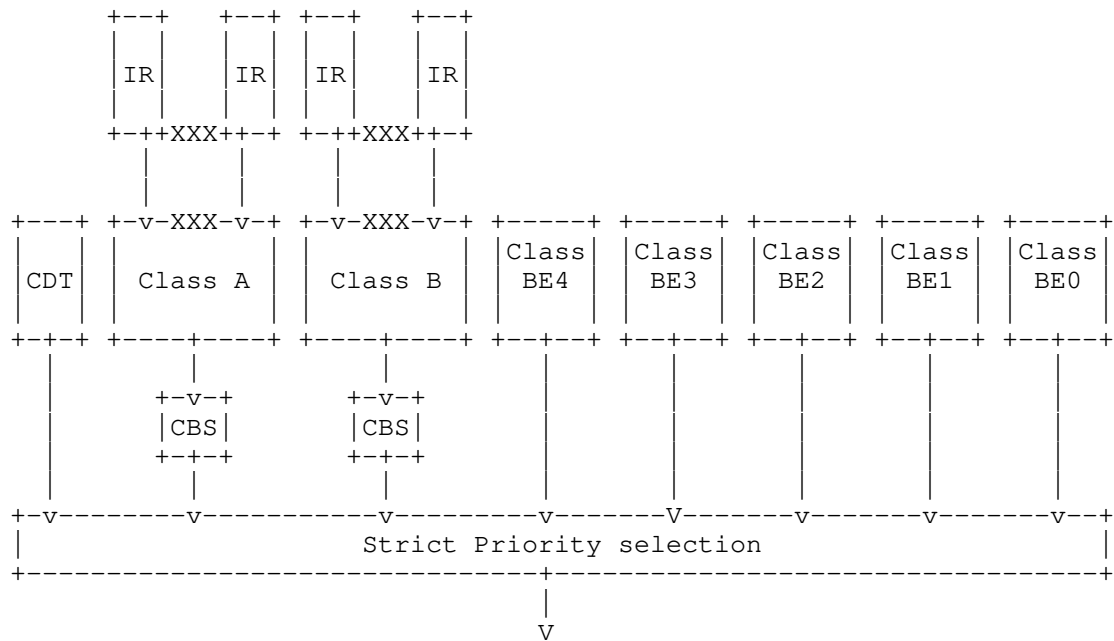


Figure 4: Architecture of a TSN node output port with interleaved regulators (IRs)

The CBFS includes two Credit-Based Shaper (CBS) subsystems, one for each class A and B. The CBS serves a packet from a class according to the available credit for that class. The credit for each class A or B increases based on the idle slope, and decreases based on the send slope, both of which are parameters of the CBS. The CDT and BE0-BE4 flows in the CBFS are served by separate FIFO subsystems. Then, packets from all flows are served by a transmission selection subsystem that serves packets from each class based on its priority. All subsystems are non-preemptive. Guarantees for AVB traffic can be provided only if CDT traffic is bounded; it is assumed that the CDT traffic has leaky bucket arrival curve with two parameters  $r_h$  as rate and  $b_h$  as bucket size, i.e., the amount of bits entering a node within a time interval  $t$  is bounded by  $r_h t + b_h$ .

Additionally, it is assumed that the AVB flows are also regulated at their source according to leaky bucket arrival curve. At the source hosts, the traffic satisfies its regulation constraint, i.e. the delay due to interleaved regulator at hosts is ignored.

At each DetNet transit node implementing an interleaved regulator, packets of multiple flows are processed in one FIFO queue; the packet at the head of the queue is regulated based on its leaky bucket



parameters; it is released at the earliest time at which this is possible without violating the constraint. The regulation parameters for a flow (leaky bucket rate and bucket size) are the same at its source and at all DetNet transit nodes along its path. A delay bound of CBFS for an AVB flow  $f$  of class A or B can be computed if the following condition holds:

sum of leaky bucket rates of all flows of this class at this node  $\leq R$ , where  $R$  is given below for every class.

If the condition holds, the delay bound is:

$$d_f = T + (b_t - L_{\min_f})/R - L_{\min_f}/c$$

where  $L_{\min_f}$  is the minimum packet length of flow  $f$ ;  $c$  is the output link transmission rate;  $b_t$  is the sum of the  $b$  term (bucket size) for all the flows having the same class as flow  $f$  at this node. Parameters  $R$  and  $T$  are calculated as follows for class A and class B, separately:

If  $f$  is of class A:

$$R = I_A (c - r_h) / c$$

$$T = L_{nA} + b_h + r_h L_n / c / (c - r_h)$$

where  $L_{nA}$  is the maximum packet length of class B and BE packets;  $L_n$  is the maximum packet length of classes A, B, and BE.

If  $f$  is of class B:

$$R = I_B (c - r_h) / c$$

$$T = (L_{BE} + L_A + L_{nA} I_A / (c_h - I_A) + b_h + r_h L_n / c) / (c - r_h)$$

where  $L_A$  is the maximum packet length of class A;  $L_{BE}$  is the maximum packet length of class BE.

Then, an end-to-end delay bound is calculated by the formula Section 4.2.2, where for  $C_{ij}$ :

$$C_{ij} = \max(d_{f'})$$

where  $f'$  is any flow that shares the same CBFS class with flow  $f$  at node  $i$  and the same interleaved regulator as flow  $f$  at node  $j$ .

More information of delay analysis in such a DetNet transit node is described in [TSNwithATS].

#### 6.4.1. Flow Admission

The delay calculation requires some information about each node. For each node, it is required to know the idle slope of CBS for each class A and B ( $I_A$  and  $I_B$ ), as well as the transmission rate of the output link ( $c$ ). Besides, it is necessary to have the information on each class, i.e. maximum packet length of classes A, B, and BE. Moreover, the leaky bucket parameters of CDT ( $r_h, b_h$ ) should be known. To admit a flow/flows, their delay requirements should be guaranteed not to be violated. As described in Section 3.1, the two problems static and dynamic are addressed separately. In either of the problems, the rate and delay should be guaranteed. Thus,

The static admission control:

The leaky bucket parameters of all flows are known, therefore, for each flow a delay bound can be calculated. The computed delay bound for every flow should not be more than its delay requirement. Moreover, the sum of the rate of each flow ( $r_f$ ) should not be more than the rate allocated to each class ( $R$ ). If these two conditions hold, the configuration is declared admissible.

The dynamic admission control:

For dynamic admission control, we allocate to every node and class A or B, static value for rate ( $R$ ) and maximum burstiness ( $b_t$ ). In addition, for every node and every class A and B, two counters are maintained:

$R_{acc}$  is equal to the sum of the leaky-bucket rates of all flows of this class already admitted at this node; At all times, we must have:

$$R_{acc} \leq R, \text{ (Eq. 1)}$$

$b_{acc}$  is equal to the sum of the bucket sizes of all flows of this class already admitted at this node; At all times, we must have:

$$b_{acc} \leq b_t. \text{ (Eq. 2)}$$

A new flow is admitted at this node, if Eqs. (1) and (2) continue to be satisfied after adding its leaky bucket rate

and bucket size to  $R_{acc}$  and  $b_{acc}$ . A flow is admitted in the network, if it is admitted at all nodes along its path. When this happens, all variables  $R_{acc}$  and  $b_{acc}$  along its path must be incremented to reflect the addition of the flow. Similarly, when a flow leaves the network, all variables  $R_{acc}$  and  $b_{acc}$  along its path must be decremented to reflect the removal of the flow.

The choice of the static values of  $R$  and  $b_t$  at all nodes and classes must be done in a prior configuration phase;  $R$  controls the bandwidth allocated to this class at this node,  $b_t$  affects the delay bound and the buffer requirement.  $R$  must satisfy the constraints given in Annex L.1 of [IEEE8021Q].

#### 6.5. IntServ

Integrated service (IntServ) is an architecture that specifies the elements to guarantee quality of service (QoS) on networks. To satisfied guaranteed service, a flow must conform to a traffic specification (T-spec), and reservation is made along a path, only if routers are able to guarantee the required bandwidth and buffer.

Consider the traffic model which conforms to token bucket regulator  $(r, b)$ , with

- o Token bucket depth  $(b)$ .
- o Token bucket rate  $(r)$ .

The traffic specification can be described as an arrival curve:

$$\alpha(t) = b + rt$$

This token bucket regulator requires that, during any time window  $t$ , the number of bit for the flow is limited by  $\alpha(t) = b + rt$ .

If resource reservation on a path is applied, IntServ model of a router can be described as a rate-latency service curve  $\beta(t)$ .

$$\beta(t) = \max(0, R(t-T))$$

It describes that bits might have to wait up to  $T$  before being served with a rate greater or equal to  $R$ .

It should be noted that, the guaranteed service rate  $R$  is a share of link's bandwidth. The choice of  $R$  is related to the specification of flows which will transmit on this node. For example, in strict priority policy, considering a flow with priority  $j$ , its share of

bandwidth may be  $R=c-\sum(r_i)$ ,  $i < j$ , where  $c$  is the link bandwidth,  $r_i$  is the token bucket rate for the flows with priority higher than  $j$ . The choice of  $T$  is also related to the specification of all the flows traversing this node. For example, in a generalized processor sharing (GPS) node,  $T = L / R + L_{\max}/c$ , where  $L$  is the maximum packet size for the flow,  $L_{\max}$  is the maximum packet size in the node across all flows. Other choice of  $R$  and  $T$  are also supported, according to the specific scheduling of the node and flows traversing this node.

As mentioned previously in this section, delay bound and backlog bound can be easily obtained by comparing arrival curve and service curve. Backlog bound, or buffer bound, is the maximum vertical derivation between curves  $\alpha(t)$  and  $\beta(t)$ , which is  $v=b+rT$ . Delay bound is the maximum horizontal derivation between curves  $\alpha(t)$  and  $\beta(t)$ , which is  $h = T+b/R$ . Graphical illustration of the IntServ model is shown in Figure 5.

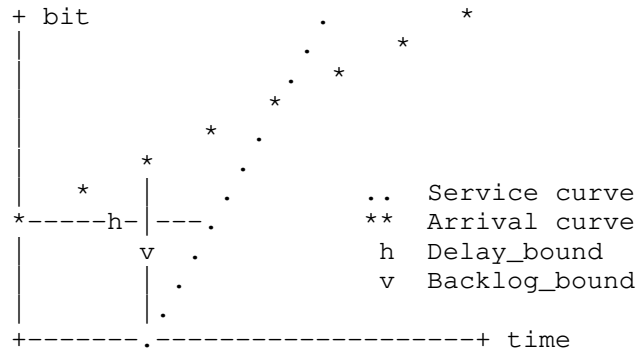


Figure 5: Computation of backlog bound and delay bound. Note that arrival and service curves are not necessary to be linear.

The output bound, or the next-hop arrival curve, is  $\alpha_{\text{out}}(t) = b + rT + rt$ , where burstiness of the flow is increased by  $rT$ , compared with the arrival curve.

We can calculate the end-to-end delay bound for a path including  $N$  nodes, among which the  $i$ -th node offers service curve  $\beta_i(t)$ ,

$$\beta_i(t) = \max(0, R_i(t-T_i)), \quad i=1, \dots, N$$

By concatenating these IntServ nodes, an end-to-end service curve can be computed as

$$\beta_{\text{e2e}}(t) = \max(0, R_{\text{e2e}}(t-T_{\text{e2e}}))$$



Figure 6 shows two DetNet transit nodes A and B, including three timelines for:

1. The output queues on port 1 in node A.
2. The input gate function ([IEEE8021Q], 8.6.5.1) that assigns packets received on port 1 of transit node B to output queues on port 2 of transit node B.
3. The output queues on port 2 of node B.

In this figure, the output ports on the two nodes are synchronized, and a new buffer starts transmitting at each tick, shown as 0, 1, 2, ... The output times shown for timelines 1 and 3 are the times at which packets are selected for output, which is the start point of the output time (1) of Figure 1. The queue assignments times on timeline 3 take place at the beginning of the queuing delay (6) of Figure 1. Time-based CQF, as described here, does not require any regulator queues. In the shown in the figure, the total time for delays 1 through 6 of Figure 1 is  $1.3T_c$ . Of course, any value is possible.

#### 7.1.1. CQF timing sequence

In general, as shown in Figure 6, the windows for buffer assignment do not align perfectly with the windows for buffer transmission. The input gates (the center timeline in Figure 6) must switch from using one buffer to using another buffer in sync with the (delayed) received data, at times offset by the dead time from the output buffer switching (the bottom timeline in Figure 6).

If the dead time  $DT$  in Figure 6 is not excessive, then it is feasible to subtract the dead time from the cycle time  $T_c$ , and use the remainder as the input window. In the example in Figure 6, packets from node A buffer a can be transferred from the input port to node B's buffer "c" during the window shown by the upper row "VVVV...". Input must cease by time = 2.0, because that is when transit node B starts transmitting the contents of buffer c. In this case, only two output buffers are in use, one filling and one outputting.

If the dead time is too large (e.g., if the delays placed the middle timeline's switching points at  $n+0.9$ , instead of  $n+0.3$ ), three buffers are used by node B. This case is shown by the lower row "VVVV..." in Figure 6. In this case, node B places the data received from node A buffer a into node B buffer d between the times 1.3 and 2.3 in Figure 6. Buffer b starts outputting at time = 2.0, while buffer d is filling. Thus, three buffers are in use, one filling, one waiting, and one emptying.

#### 7.1.2. Dead time computation

The time for switching input packet buffer assignments is equal to the minimum possible offset from transmission selection in node A to buffer assignment in node B, which is the sum of the minimum values for all of the delays 1 through 5 in Figure 1 (the queue-to-queue delay). All packets must be received and assigned to an output buffer before the next switching point, which means that all must be transmitted in time for them to arrive at buffer assignment even if worst-case (longest delay) is encountered for the queue-to-queue delay. Thus, the minimum dead time for the 3-buffer case is the sum of the worst-case variation in the queue-to-queue delay, plus the worst-case difference between the two transit nodes' buffer switching clocks.

For the 2-buffer case, we must add the offset (shown as "DT" in Figure 6) from the end of node B's output switch to the end of node B's input switch.

#### 7.1.3. Tc computation

Given the dead time DT, there remains a transmit window of  $(Tc - DT - Int)$ . The DT was explained in (Section 7.1.2). "Int" is the worst-case interference with the start of transmission, when the output buffers switch, caused by lower-priority traffic. This is equal to one worst-case transmission time, which means that the size of the packets in all lower-priority queues must be bounded. If Ethernet preemption ([IEEE8023] clause 99) is employed for lower-priority queues, then this worst-case interference is reduced to the size of the largest unfragmentable Ethernet frame.

The bandwidth requirement of any given DetNet flow has to be translated to CQF terms, in order to determine whether that flow can be accommodated at each port. A flow has to be characterized as using a maximum number of bit times on the wire per cycle time Tc. For Ethernet, for example ([IEEE8023]), this includes the preamble (8 bytes), destination MAC address through CRC (minimum 64 bytes) and the inter-packet gap (12 bytes). The total bit times per cycle Tc required by all of the DetNet flows passing through a given port cannot exceed the available transmit window  $(Tc - DT - Int)$ .

#### 7.1.4. CQF latency calculation

The per-hop latency is trivially determined by the wire delay plus the queuing delay. Since the wire delay is either absorbed into the queueing delay (dead time is small and two buffers are used) or padded out to a whole cycle time Tc (three buffers are used) the per-

hop latency is always an integral number of cycle times  $T_c$ , with a latency variation at the output of the final hop of  $T_c$ .

Ingress conditioning may be required if the source of a DetNet flow does not, itself, employ CQF. See Section 7.1.6.

#### 7.1.5. CQF parameterization

The transmit window for a given DetNet transit node running CQF, for example the transmit window for node A in Figure 6, depends on the interference (Int in section Section 7.1.3), the value of  $T_c$ , and the dead time required by the following node B. The size of the transmit window determines how many total bits can be reserved per period  $T_c$  by DetNet flows.

Part of the dead time derives from delays and delay variations such as output delay (1) and link delay (2) and preemption delay (3) of Figure 1, all of which are known to node A. However, the dead time also depends on the processing delay (4) of node B and the upon whether node B is using 2 or 3 output buffers, which is not necessarily known to node A.

The information in DetNet transit node B necessary to compute the dead time to be observed by transit node A must be known to the entity responsible for making reservation decisions, whether that is node A itself, or a central controller. A decision can be made, by the controller or by the node, whether to use the dead time and two buffers, in order to reduce the per-hop latency by one cycle time, or to use three buffers and eliminate the dead time and increase the total allocable bandwidth.

If the packet sizes of a DetNet flow are variable, or perhaps even unknown beyond the imposition of a maximum size, then some degree of overprovisioning is required. The measurement used to allocate bandwidth to a given DetNet flow is bit times in one cycle time  $T_c$ . Therefore, one extra maximum packet time (less one bit) has to be allocated to a flow per cycle time  $T_c$  in order to ensure that, no matter what mix of packet sizes are presented, the flow will get its guaranteed latency.

#### 7.1.6. Ingress conditioning for CQF

Assuming that a DetNet domain is using CQF, it is always possible that the previous node (or sender) may not support the queuing method of CQF, or may support CQF but not use the same configuration in accordance of the current DetNet domain. In this case, ingress conditioning is helpful to shape the flow according to the TSPEC in



the current DetNet domain and transmission cycle of CQF, thus control the burstiness and reduce overprovisioning.

A DetNet node running CQF satisfies that a maximum number of `max_number_packet_per_cycle` packets, each with length no larger than `max_packet_size`, can be transmitted during a CQF cycle  $T_c$ . Also, the dead time  $D_t$ , during which the former node cannot transmit to the later node (See Section 7.1.1), should be considered. Here we use the notation `max_number_packet_per_cycle`, `max_packet_size` to describe the maximum amount of transmitting data during the available transmit window  $T_c - D_t$ .

An ingress conditioner typically consists with a FIFO queue with an output regulator. Every incoming packet enters the FIFO queue, which passes it on if and only if the packet conforms to the CQF requirement. For this purpose, two criteria below is suggested to follow.

- o The incoming flow's average rate would be no more than the average output queue rate at ingress conditioner, to avoid overflow and congestion loss.
- o The queueing buffer at ingress conditioner is suggested to be large enough, to cover burstiness and jitter of bit rate from the previous node (or sender).

The output regulator controls the transmitting of packets, with credit function for instance. At the start of a CQF cycle, credit is set to the maximum bits to transmit in a cycle, `max_bit_per_cycle = max_number_packet_per_cycle * max_packet_size`. When a packet is transmitted from the node, the credit is reduced by `length(packet)`. The operation of the output regulator can be described as below.

```

credit = max_bit_per_cycle; % Initial credit
t = 0; % Initial time offset within a cycle

outputRegulate(packet){
if (t>=0 && t< Tc-Dt) % during a transmit window
{
    if ~isEmpty(queue) % if queue is not empty
    {
        if credit >= length(packet);
        {
            dequeue(packet);
            credit = credit - length(packet);
        }
    }
}
elseif (t >= Tc) % when a cycle end, reset t and credit.
{
    t = t - Tc; % reset t to [0, Tc-Dt)
    credit = max_bit_per_cycle; % credit is refilled
}}

```

Other instance of output regulator may also meet the CQF requirement.

#### 7.1.1.7. CQF ingress conditioning timing model

Consider the input traffic conforms to variable bit rate (VBR) constraint  $(p, M, r, b)$ , which can be modeled as arrival curve:

$$\alpha(t) = \min(pt+M, rt+b)$$

where  $p$  is the peak rate,  $M$  is the maximum size of a packet,  $(r, b)$  stand for token bucket rate and burst. Note that, if  $N$  flows enter the ingress conditioner, each flow  $f_i$  conforms to token bucket  $(r_i, b_i)$ , the arrival curve parameters are of the superposition form:  $r = \sum(r_i)$ ,  $b = \sum(b_i)$ ,  $i=1, \dots, N$ .

Ingress conditioner, which transmit packets as soon as possible if conformance is satisfied, does not increase worst-case queuing latency, if we consider the latency from the input of ingress conditioner to the first output of CQF node. The CQF node, which transmits at most  $\text{max\_bit\_per\_cycle}$  bits during a cycle  $T_c$ , offers a service curve

$$\beta(t) = \text{max\_bit\_per\_cycle} * \text{floor}(t-Dt) + \min(p * \text{mod}(t-Dt, T_c), \text{max\_bit\_per\_cycle})$$

where  $Dt$  is the dead time (see Section 7.1.1),  $p$  is the peak rate,  $\text{max\_bit\_per\_cycle} = \text{max\_number\_packet\_per\_cycle} * \text{max\_packet\_size}$  is

the maximum transmitting data in a CQF cycle,  $\text{floor}(x)$  calculates the nearest integer less than or equal to  $x$ , and  $\text{mod}(x,y)=x-\text{floor}(x/y)$ .

According to network calculus, the worst-case queuing delay for ingress conditioner and CQF node, denoted  $\text{delay\_bound}$ , is the maximum horizontal distance between arrival curve and service curve.

Specifically, in general assumptions, ingress conditioner's average output rate  $r_2 = \text{max\_bit\_per\_cycle}/T_c$  is always no less than input traffic rate  $r$  (or else congestion loss would happen). In such conditions, the queuing delay up-bound for ingress conditioner and first CQF node is derived as  $\text{ingress\_delay\_bound} = \max(t : \beta(t) = (pb-rm) / (p-r))$ . Therefore, considering a path traverses ingress conditioner and  $H$  CQF nodes, the total delay up-bound is  $H \cdot T_c + \text{ingress\_delay\_bound}$ .

## 7.2. Time-scheduled queuing

[IEEE8021Q] section 8.6.8.4 specifies a time-aware queue-draining procedure for transmission selection at egress port of a DetNet transit node, which supports up to eight traffic classes. Each traffic class has a separate queue, frame transmission from each queue is allowed or prevented by a time gate. This time gate controlled scheduling allows time-sensitive traffic classes to transmit on dedicate time slots. Within the time slots, the transmitting flows can be granted exclusive use of the transmission medium. Generally, this time-aware scheduling is a layer 2 time division multiplexing (TDM) technique.

Consider the static configuration of a deterministic network. To provide end-to-end latency guaranteed service, network nodes can support time-based behavior, which is determined by gate control list (GCL). GCL defines the gate operation, in open or closed state, with associated timing for each traffic class queue. A time slice with gate state "open" is called transmission window. The time-based traffic scheduling must be coordinated among the DetNet transit nodes along the path from sender to receiver, to control the transmission of time-sensitive traffic.

Ideally all network devices are time synchronized and static GCL configurations on all devices along the routed path are coordinated to ensure that length of transmission window fits the assigned frames, and no two time windows for DetNet traffic on the same port overlap. (DetNet flows' windows can overlap with best-effort windows, so that unused DetNet bandwidth is available to best-effort traffic.) The processing delay, link delay and output delay in transmitting are considered in GCL computation. Transmission window for a certain flow may require that a time offset on consecutive hops

be selected to reduce queueing delay as much as possible. In this case, TSN/DetNet frames transmit at the assigned transmission window at every node through the routed path, with zero congestion loss and bounded end-to-end latency. Then, the worst-case end-to-end latency of the flow can be derived from GCL configuration. For a TSN or DetNet frame, denote the transmission window on last hop closes at `gate_close_time_last_hop`. Assuming talker supports scheduled traffic behavior, it starts the transmission at `gate_open_time_on_talker`. Then worst case end-to-end delay of this flow is bounded by `gate_close_time_last_hop - gate_open_time_on_talker + link_delay_last_hop`.

It should be noted that scheduled traffic service relies on a synchronized network and coordinated GCL configuration. Synthesis of GCL on multiple nodes in network is a scheduling problem considering all TSN/DetNet flows traversing the network, which is a non-deterministic polynomial-time hard (NP-hard) problem. Also, at this writing, scheduled traffic service supports no more than eight traffic classes, typically using up to seven priority classes and at least one best effort class.

## 8. Parameters for the bounded latency model

The use of the TSPEC parameters defined in [RFC2212] and related documents for IntServ are well-established and adequate for DetNet purposes. The parameterization used by [IEEE8021Q] are somewhat different, as discussed above (Section 7.1.6). These parameters are maximum number of frames per interval, interval size, and maximum frame size. They are more suitable for the physical determination of compliance by a sender than for resource reservation purposes.

## 9. References

### 9.1. Normative References

- [I-D.ietf-detnet-architecture]  
Finn, N., Thubert, P., Varga, B., and J. Farkas,  
"Deterministic Networking Architecture", draft-ietf-detnet-architecture-08 (work in progress), September 2018.
- [I-D.ietf-detnet-dp-sol-ip]  
Korhonen, J. and B. Varga, "DetNet IP Data Plane Encapsulation", draft-ietf-detnet-dp-sol-ip-00 (work in progress), July 2018.

- [I-D.ietf-detnet-dp-sol-mpls]  
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane Encapsulation", draft-ietf-detnet-dp-sol-mpls-00 (work in progress), July 2018.
- [I-D.ietf-detnet-use-cases]  
Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-20 (work in progress), December 2018.
- [RFC2212] Shenker, S., Partridge, C., and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, DOI 10.17487/RFC2212, September 1997, <<https://www.rfc-editor.org/info/rfc2212>>.
- [RFC6658] Bryant, S., Ed., Martini, L., Swallow, G., and A. Malis, "Packet Pseudowire Encapsulation over an MPLS PSN", RFC 6658, DOI 10.17487/RFC6658, July 2012, <<https://www.rfc-editor.org/info/rfc6658>>.
- [RFC7806] Baker, F. and R. Pan, "On Queuing, Marking, and Dropping", RFC 7806, DOI 10.17487/RFC7806, April 2016, <<https://www.rfc-editor.org/info/rfc7806>>.

## 9.2. Informative References

- [bennett2002delay]  
J.C.R. Bennett, K. Benson, A. Charny, W.F. Courtney, and J.-Y. Le Boudec, "Delay Jitter Bounds and Packet Scale Rate Guarantee for Expedited Forwarding", <<https://dl.acm.org/citation.cfm?id=581870>>.
- [charny2000delay]  
A. Charny and J.-Y. Le Boudec, "Delay Bounds in a Network with Aggregate Scheduling", <[https://link.springer.com/chapter/10.1007/3-540-39939-9\\_1](https://link.springer.com/chapter/10.1007/3-540-39939-9_1)>.
- [IEEE8021Q]  
IEEE 802.1, "IEEE Std 802.1Q-2018: IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks", 2018, <<http://ieeexplore.ieee.org/document/8403927>>.
- [IEEE8021Qcr]  
IEEE 802.1, "IEEE P802.1Qcr: IEEE Draft Standard for Local and metropolitan area networks - Bridges and Bridged Networks - Amendment: Asynchronous Traffic Shaping", 2017, <<http://www.ieee802.org/1/files/private/cr-drafts/>>.

- [IEEE8021TSN]  
IEEE 802.1, "IEEE 802.1 Time-Sensitive Networking (TSN) Task Group", <<http://www.ieee802.org/1/>>.
- [IEEE8023]  
IEEE 802.3, "IEEE Std 802.3-2018: IEEE Standard for Ethernet", 2018,  
<<http://ieeexplore.ieee.org/document/8457469>>.
- [le\_boudec\_theory\_2018]  
J.-Y. Le Boudec, "A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators", <<http://arxiv.org/abs/1801.08477/>>.
- [NetCalBook]  
Le Boudec, Jean-Yves, and Patrick Thiran, "Network calculus: a theory of deterministic queuing systems for the internet", 2001, <<https://arxiv.org/abs/1804.10608/>>.
- [Specht2016UBS]  
J. Specht and S. Samii, "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks",  
<<https://ieeexplore.ieee.org/abstract/document/7557870>>.
- [TSNwithATS]  
E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, "End-to-end Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping",  
<<https://arxiv.org/abs/1804.10608/>>.

## Authors' Addresses

Norman Finn  
Huawei Technologies Co. Ltd  
3101 Rio Way  
Spring Valley, California 91977  
US

Phone: +1 925 980 6430  
Email: [norman.finn@mail01.huawei.com](mailto:norman.finn@mail01.huawei.com)

Jean-Yves Le Boudec  
EPFL  
IC Station 14  
Lausanne EPFL 1015  
Switzerland

Email: [jean-yves.leboudec@epfl.ch](mailto:jean-yves.leboudec@epfl.ch)

Ehsan Mohammadpour  
EPFL  
IC Station 14  
Lausanne EPFL 1015  
Switzerland

Email: [ehsan.mohammadpour@epfl.ch](mailto:ehsan.mohammadpour@epfl.ch)

Jiayi Zhang  
Huawei Technologies Co. Ltd  
Q22, No.156 Beiqing Road  
Beijing 100095  
China

Email: [zhangjiayi11@huawei.com](mailto:zhangjiayi11@huawei.com)

Balazs Varga  
Ericsson  
Konyves Kalman krt. 11/B  
Budapest 1097  
Hungary

Email: [balazs.a.varga@ericsson.com](mailto:balazs.a.varga@ericsson.com)

Janos Farkas  
Ericsson  
Konyves Kalman krt. 11/B  
Budapest 1097  
Hungary

Email: [janos.farkas@ericsson.com](mailto:janos.farkas@ericsson.com)

Network Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: September 12, 2019

X. Geng  
M. Mach  
Huawei  
March 11, 2019

DetNet SRv6 Data Plane Encapsulation  
draft-geng-detnet-dp-sol-srv6-00

Abstract

This document specifies Deterministic Networking data plane operation for SRv6 encapsulated user data.

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 RFC 2119 [RFC2119].

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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
2. Terminology and Conventions . . . . .	3
2.1. Terminology . . . . .	3
2.2. Conventions . . . . .	3
3. SRv6 DetNet Data Plane Overview . . . . .	4
3.1. SRv6 DetNet Data Plane Layers . . . . .	4
3.2. SRv6 DetNet Data Plane Scenarios . . . . .	4
4. SRv6 DetNet Data Plane Solution Considerations . . . . .	6
5. SRv6 DetNet Data Plane Solution for Service Sub-layer . . . . .	7
5.1. TLV Based SRv6 Data Plane Solution . . . . .	7
5.1.1. Encapsulation . . . . .	7
5.1.2. Functions . . . . .	9
5.2. SID Based SRv6 Data Plane Solution . . . . .	10
5.2.1. Encapsulation . . . . .	10
5.2.2. Functions . . . . .	11
5.3. DetNet SID Based SRv6 Data Plane Solution . . . . .	12
5.3.1. Encapsulation . . . . .	12
5.3.2. Functions . . . . .	13
5.4. DetNet SRH Based SRv6 Data Plane Solution . . . . .	13
5.4.1. Encapsulation . . . . .	13
5.4.2. Functions . . . . .	14
5.5. MPLS Based SRv6 Data Plane Solution . . . . .	14
6. SRv6 DetNet Data Plane Solution for Transport Sub-layer . . . . .	15
7. IANA Considerations . . . . .	15
8. Security Considerations . . . . .	15
9. Acknowledgements . . . . .	15
10. Normative References . . . . .	15
Authors' Addresses . . . . .	16

## 1. Introduction

Deterministic Networking(DetNet) provides a capability to carry specified data flows with extremely low data loss rates and bounded latency within a network domain. DetNet is enabled by a group of technologies, such as resource allocation, service protection and explicit routes.([I-D.ietf-detnet-architecture])

Segment Routing(SR) leverages the source routing paradigm. A ingress node steers a packet through an ordered list of instructions, called "segments". SR can be applied over IPv6 data plane using Routing Extension Header(SRH). Besides routing, the segment of SRv6 can indicate functions which are executed locally in the node where they

are defined. SRv6 network programming makes it convenient to add sophisticated operations in the network. ([RFC8402])

This document describes how to implement DetNet with SRv6. It can provide : 1. Source routing, which can steer the DetNet flows go through the network according to an explicit route with allocated resource; 2. Network programming, which can give packet instructions in some special nodes (even all the nodes) along the path to guarantee service protection and congestion protection. DetNet SRv6 encapsulation and new SRv6 functions for DetNet are defined in this document.

Control plane and OAM are not in the scope of this document.

## 2. Terminology and Conventions

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 [RFC2119].

### 2.1. Terminology

Terminologies for DetNet go along with the definition in [I-D.ietf-detnet-architecture]. Other terminologies are defined as follows:

- o NH: The IPv6 next-header field.
- o SID: A Segment Identifier which represents a specific segment in a segment routing domain([RFC8402]).
- o SRH: The Segment Routing Header ([I-D.ietf-6man-segment-routing-header]).

### 2.2. Conventions

Conventions in the document are defined as follows:

- o NH=SRH means that NH is 43 with routing type 4.
- o A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.
- o SRH[SL] represents the SID pointed by the SL field in the first SRH. In our example, SRH[2] represents S1, SRH[1] represents S2 and SRH[0] represents S3.

- o (SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

IPv6 header with source and destination addresses SA and DA respectively, and next-header SRH, with SID list <S1, S2, S3> with SegmentsLeft = SL

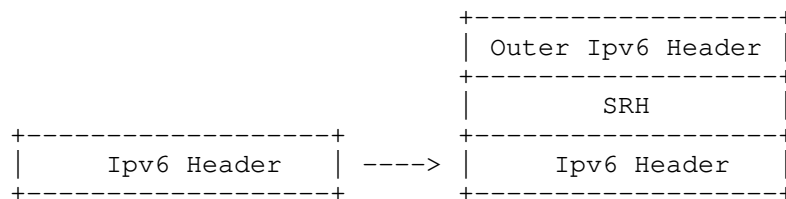
The payload of the packet is not represented

(S3, S2, S1; SL) represents the same SID list as <S1, S2, S3>, but encoded in the SRH format where the rightmost SID in the SRH is the first SID and the leftmost SID in the SRH is the last SID

### 3. SRv6 DetNet Data Plane Overview

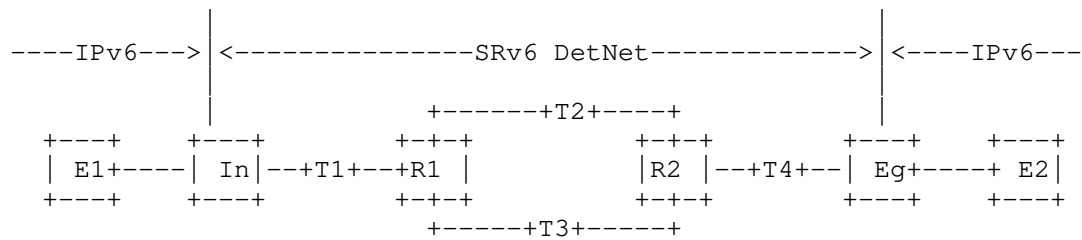
#### 3.1. SRv6 DetNet Data Plane Layers

[I-D.ietf-detnet-architecture]decomposes the DetNet data plane into two sub-layers: service sub-layer and transport sub-layer. Different from DetNet MPLS data plane solution, which uses DetNet Control Word(d-CW) and S-Label to support service sub-layer and uses T-Label to support transport sub-layer, no explicit sub-layer division can be found in SRv6 data plane. A classical SRv6 DetNet data plane solution is showed in the picture below:



The outer IPv6 Header with SRH is used for carrying DetNet flows. Traffic Engineering is programmed in the segment list of SRH, and other functions and arguments for service protection (packet replication, elimination and ordering) and congestion control (packet queuing and forwarding) are also defined in SRH.

#### 3.2. SRv6 DetNet Data Plane Scenarios



The figure above shows that an IPv6 flow is sent out from the end station: E1. The packet of the flow is encapsulated as a DetNet SRv6 packet in the Ingress(In) and transported through an SRv6 DetNet domain. In the Egress(Eg), the upper IPv6 header with SRH of the packet is popped, and the packet is transmitted to the destination(E2).

The DetNet packet processing is as follows:

#### Ingress:

Insert SRv6 Policy, which can steer the packet from Ingress to Relay Node 1

Flow Identification and Sequence Number are carried in SRH

#### Relay Node 1(Replication Node):

Replicate the payload and IPv6 Header with SRH

Binding two different SRv6 Policy respectively to the original packet and the replicated packet, which can steer the packet from Relay Node 1 to Relay Node 2 through two tunnels

#### Relay Node 2(Elimination Node):

Eliminate the redundant packets

Binding a new SRv6 Policy to the survival packet, which can steer the packet from Relay Node 2 to Egress.

#### Egress:

Decapsulate the upper Ipv6 header

Send the packet to the End Station 2

The DetNet packet encapsulation is as follows:

End Station1 out : (E1,E2)

Ingress out : (In, T1) (R1,T1,SL=2) (E1,E2)

Transit Node1 out : (In, R1) (R1,T1,SL=1) (E1,E2)

Relay Node1 out : (R2, R1) (R2,T2,SL=2) (E1,E2) / (R2, R1) (R2,T3,SL=2) (E1,E2)

Transit Node2 out : (R2, R1) (R2,T2,SL=1) (E1,E2)

Transit Node3 out : (R2, R1) (R2,T3,SL=1) (E1,E2)

Relay Node2 out : (Eg, R2) (Eg,T4,SL=2) (E1,E2)

Transit Node4 out : (Eg, R2) (Eg,T4,SL=1) (E1,E2)

Egress out : (E1,E2)

#### 4. SRv6 DetNet Data Plane Solution Considerations

To carry DetNet over SRv6, the following elements are required:

1. A method of identifying the SRv6 payload type;
2. A suitable explicit route to deliver the DetNet flow ;
3. A method of indicating packet processing, such as PREOF;
4. A method of identifying the DetNet flow;
5. A method of carrying DetNet sequence number;
6. A method of carrying queuing and forwarding indication to do congestion protection;

In this design, DetNet flows are encapsulated with SRH in the Ingress Node. The SR policy in the SRH steers the DetNet flow along a selected path. The explicit route allocated to a DetNet flow, which protect it from temporary interruptions caused by the convergence of routing, is encoded within the SID list of a SR policy. The network device inside the DetNet domain forwards the packet according to IPv6 Destination Address(DA), and the IPv6 DA is updated with the SID list.

With SRv6 network programming, the SID list can also give instruments representing a function to be called at the node in the DetNet domain. Therefore DetNet specific functions defined in

[I-D.ietf-detnet-architecture], corresponding to local packet processing in the network, can also be implemented by SRv6. New functions associated with SIDs for DetNet are defined in this document.

This document describes how DetNet flows are encapsulated/identified, and how functions of Packet Replication/Elimination/Ordering are implemented in an SRv6 domain. Congestion protection is also in the scope of this document.

Editor: This version only covers the functions of service protection and the congestion protection considerations will be added in the following versions.

## 5. SRv6 DetNet Data Plane Solution for Service Sub-layer

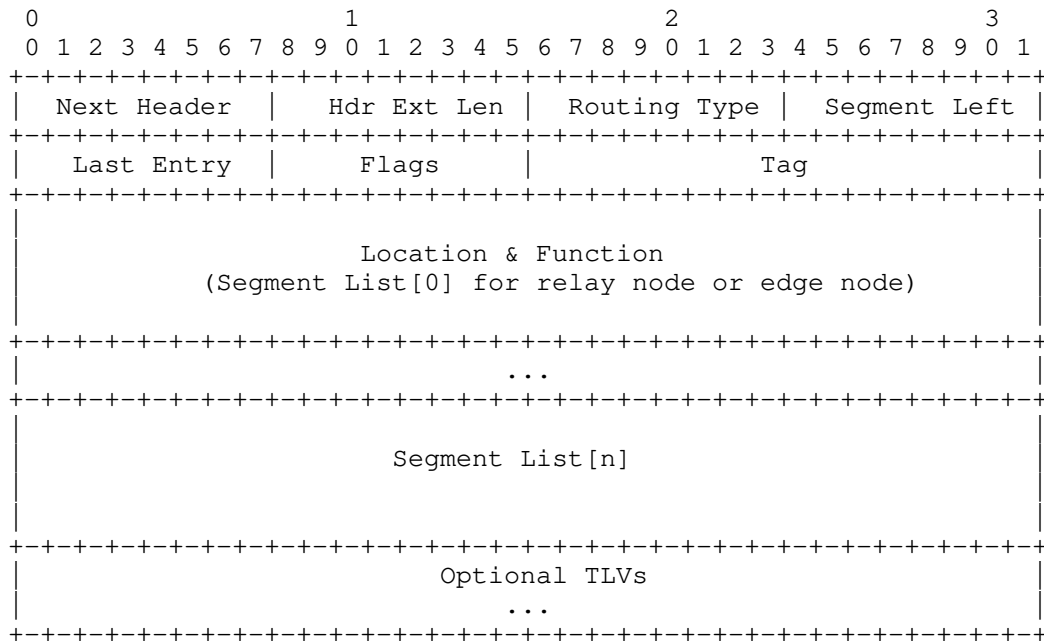
This section defines options of SRv6 data plane solution to support DetNet Service Sub-layer.

### 5.1. TLV Based SRv6 Data Plane Solution

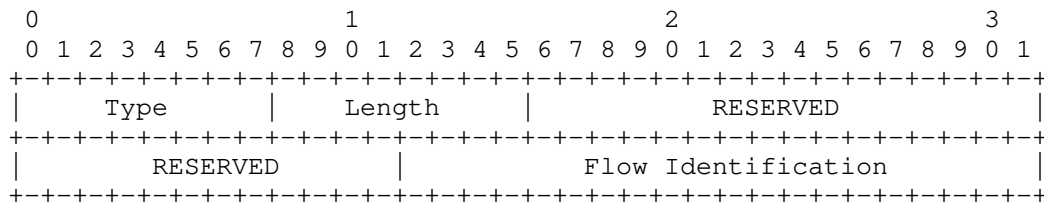
#### 5.1.1. Encapsulation

An SRv6 Segment is a 128-bit value. SID is used as a shorter reference for "SRv6 Segment". SRv6 SID can also be represented as LOC:FUNCT, where LOC, abbreviated for "LOCATION", directs the explicit route, FUNCT, abbreviated for "FUNCTION", directs the packet processing in the local node ([I-D.filsfils-spring-srv6-network-programming]).

The SRH for DetNet in the outer IPv6 header is showed as follows:

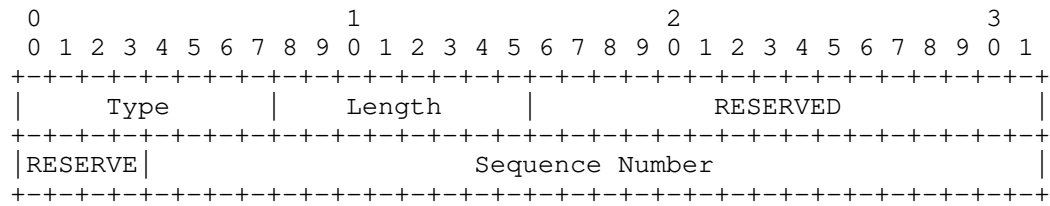


Two new TLVs are defined to support DetNet service protection. DetNet Flow Identification TLV is used to uniquely identify a DetNet flow in an SRv6 DetNet node. DetNet sequence number is used to dis crime packets in the same DetNet flow. They are defined as follows:



where:

- o Type: 8bits, to be assigned by IANA.
- o Length: 8.
- o RESERVED: 28 bits, MUST be 0 on transmission and ignored on receipt.
- o Flow Identification: 20 bits, which is used for identifying DetNet flow.



where:

- o Type: 8 bits, to be assigned by IANA.
- o Length: 8.
- o RESERVED: 20 bits. MUST be 0 on transmission and ignored on receipt.
- o Sequence Number: 28 bits, which is used for indicating sequence number of a DetNet flow.

#### 5.1.2. Functions

New SID functions are defined as follows:

##### 5.1.2.1. End. B.Replicationreserve the value of argument field(Inherited argument)of segment[0] of SRH n: Packet Replication Function

1. IF NH=SRH & SL>0 THEN
2. do not decrement SL nor update the IPv6 DA with SRH[SL]
3. reserve the value of DetNet TLVs of SRH
4. add the DetNet TLVs into SRH'1 and SRH'2
5. pop the SRH
6. replicate the packet into two packets: packet'1, packet'2
7. insert SRH'1 to packet'1
8. insert SRH'2 to packet'2
9. set the IPv6 DA of packet'1 to the first segment of the SRv6 Policy of SRH'1
10. set the IPv6 DA of packet'2 to the first segment of the SRv6 Policy of SRH'2



11. ELSE

12. drop the packet

#### 5.1.2.2. End. B. Elimination: Packet Elimination Function

1. IF NH=SRH & SL>0 & "the packet is not a redundant packet" THEN

2. do not decrement SL nor update the IPv6 DA with SRH[SL]

3. reserve the value of DetNet TLVs of SRH

4. add the DetNet TLVs into SRH'

5. pop the SRH

6. insert SRH'

7. set the IPv6 DA to the first segment of the SRv6 Policy

8. ELSE

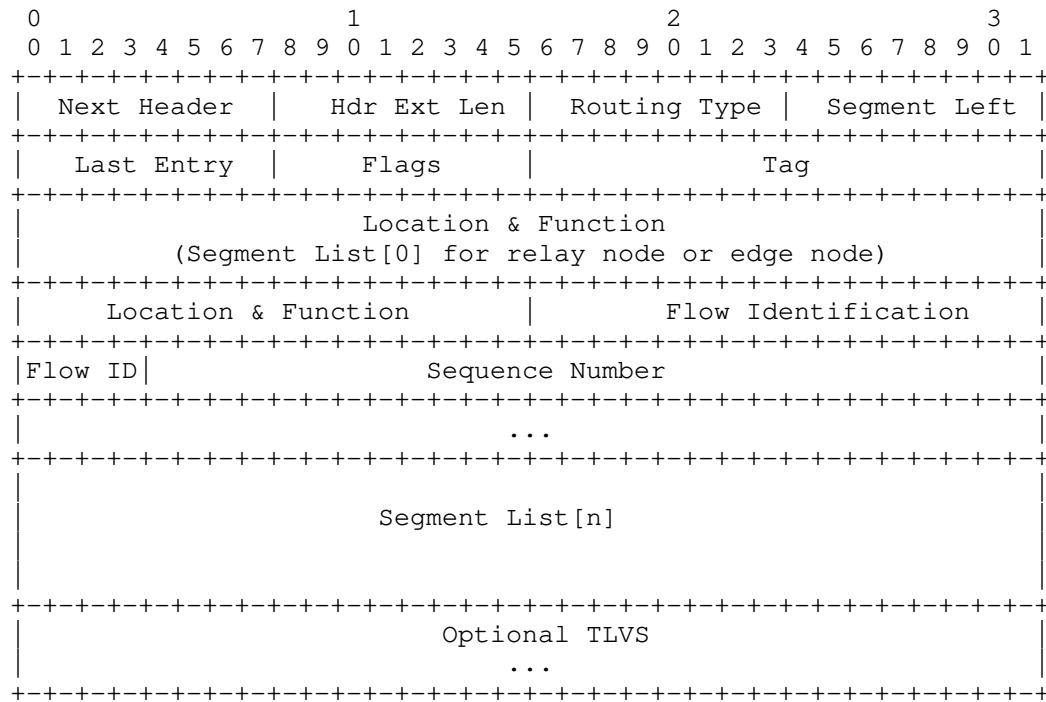
9. drop the packet

#### 5.2. SID Based SRv6 Data Plane Solution

##### 5.2.1. Encapsulation

SRv6 SID can be represented as LOC:FUNCT:ARG::, where LOC, abbreviated for "LOCATION", directs the explicit route, FUNCT, abbreviated for "FUNCTION", directs the packet processing in the local node, and ARG, abbreviated for "ARGUMENTS", provides the additional arguments for the functions. New SID functions for DetNet is defined in section 5.2.2.

The SRH for DetNet in the outer IPv6 header is showed as follows:



where:

- o LOCATION&FUNCTION: the 80 most significant bits that are used for routing;
- o FLOW IDENTIFICATION: 20 bits, which is used for DetNet flow identification in the DetNet relay node;
- o SEQUENCE NUMBER : 28 bits, which are used for dis crime packets in the same DetNet flow;

#### 5.2.2. Functions

New SID functions are defined as follows:

##### 5.2.2.1. End. B.Replication: Packet Replication Function

1. IF NH=SRH & SL>0 THEN
2. do not decrement SL nor update the IPv6 DA with SRH[SL]
3. reserve the value of argument field(Inherited argument)of segment[0] of SRH

4. write the inherited arguments into the argument field of segment[0] of SRH'1 and SRH'2
5. pop the SRH
6. replicate the packet into two packets: packet'1, packet'2
7. insert SRH'1 to packet'1
8. insert SRH'2 to packet'2
9. set the IPv6 DA of packet'1 to the first segment of the SRv6 Policy of SRH'1
10. set the IPv6 DA of packet'2 to the first segment of the SRv6 Policy of SRH'2
11. ELSE
12. drop the packet

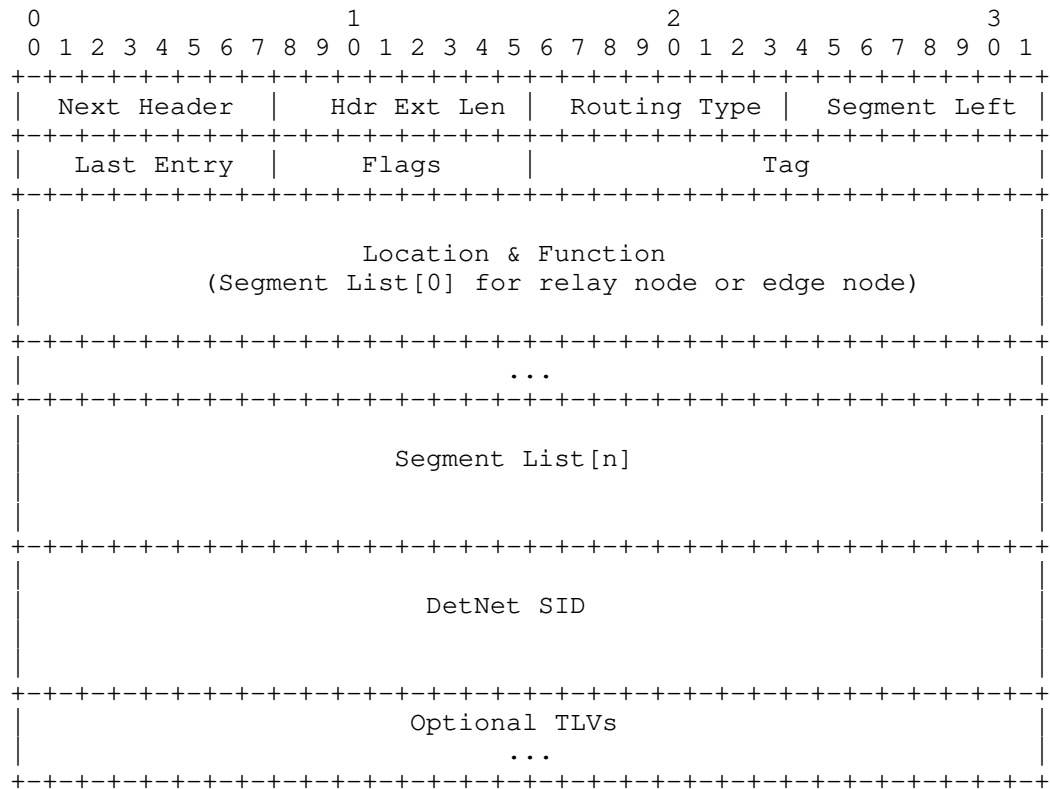
#### 5.2.2.2. End. B. Elimination: Packet Elimination Function

1. IF NH=SRH & SL>0 & "the packet is not a redundant packet" THEN
2. do not decrement SL nor update the IPv6 DA with SRH[SL]
3. write the inherited arguments into the argument field of segment[0] of SRH'
4. pop the SRH
5. insert SRH'
6. set the IPv6 DA to the first segment of the SRv6 Policy
7. ELSE
8. drop the packet

#### 5.3. DetNet SID Based SRv6 Data Plane Solution

##### 5.3.1. Encapsulation

A non-forwarding DetNet SID is defined to carry Flow Identification and Sequence Number.



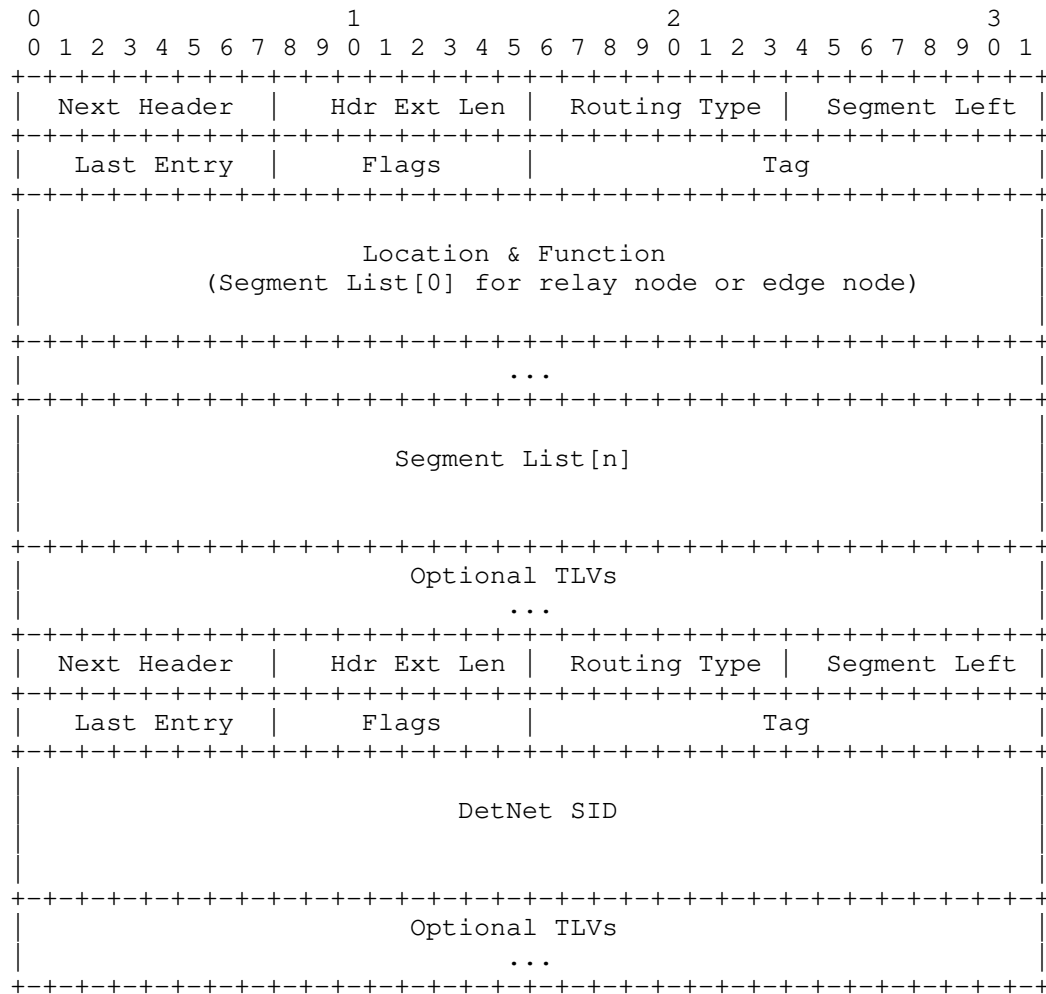
### 5.3.2. Functions

TBD

## 5.4. DetNet SRH Based SRv6 Data Plane Solution

### 5.4.1. Encapsulation

A New SRH is defined to carry Flow Identification and Sequence Number.

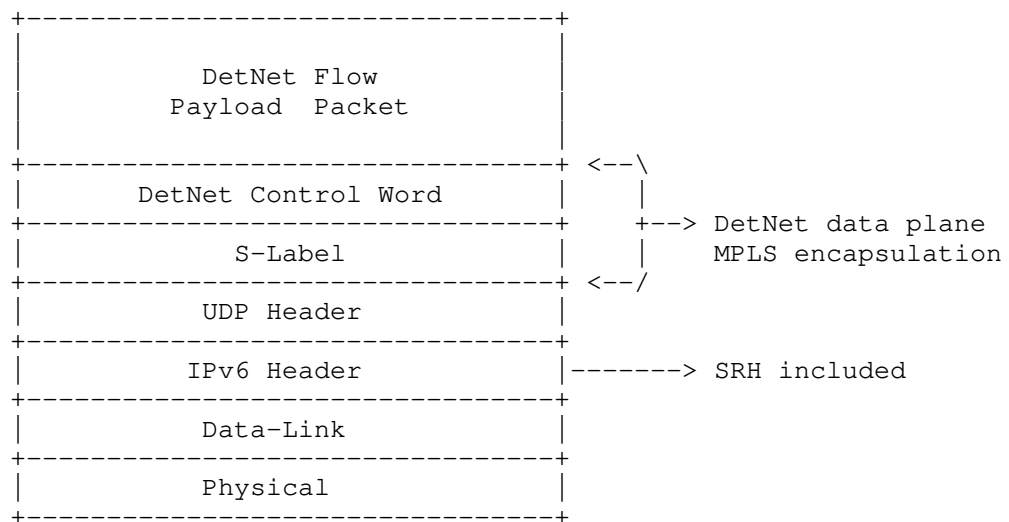


#### 5.4.2. Functions

TBD

#### 5.5. MPLS Based SRv6 Data Plane Solution

SRH can be part of IPv6 Header in the picture below, and no protocol extensions are needed in SRH. The structure keeps the same as the definition in [I-D.ietf-detnet-dp-sol-mpls] :



## 6. SRv6 DetNet Data Plane Solution for Transport Sub-layer

TBD

## 7. IANA Considerations

This document makes no request of IANA.

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

## 8. Security Considerations

TBD

## 9. Acknowledgements

Thank you for valuable comments from James Guichard and Andrew Mails.

## 10. Normative References

- [I-D.filsfils-spring-srv6-network-programming]  
 Filsfils, C., Camarillo, P., Leddy, J.,  
 daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6  
 Network Programming", draft-filsfils-spring-srv6-network-  
 programming-07 (work in progress), February 2019.

- [I-D.ietf-6man-segment-routing-header]  
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and  
d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header  
(SRH)", draft-ietf-6man-segment-routing-header-16 (work in  
progress), February 2019.
- [I-D.ietf-detnet-architecture]  
Finn, N., Thubert, P., Varga, B., and J. Farkas,  
"Deterministic Networking Architecture", draft-ietf-  
detnet-architecture-11 (work in progress), February 2019.
- [I-D.ietf-detnet-dp-sol-mpls]  
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane  
Encapsulation", draft-ietf-detnet-dp-sol-mpls-01 (work in  
progress), October 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L.,  
Decraene, B., Litkowski, S., and R. Shakir, "Segment  
Routing Architecture", RFC 8402, DOI 10.17487/RFC8402,  
July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.

## Authors' Addresses

Xuesong Geng  
Huawei

Email: [gengxuesong@huawei.com](mailto:gengxuesong@huawei.com)

Mach(Guoyi) Chen  
Huawei

Email: [mach.chen@huawei.com](mailto:mach.chen@huawei.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 9, 2019

L. Geng  
China Mobile  
L. Qiang  
Huawei Technologies  
T. Eckert  
Huawei  
March 8, 2019

Technical Requirements of Bounded Latency Forwarding  
draft-geng-detnet-requirements-bounded-latency-01

Abstract

This document analyses the technical requirements that Layer 3 bounded latency forwarding scheme should satisfy.

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 September 9, 2019.

Copyright Notice

Copyright (c) 2019 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 . . . . .	2
1.2. Terminology & Abbreviations . . . . .	3
2. Tolerance of Time Deviation . . . . .	3
3. Long Link Propagation Delay . . . . .	3
4. Massive Dynamic Flows . . . . .	4
5. IANA Considerations . . . . .	4
6. Security Considerations . . . . .	4
7. Acknowledgements . . . . .	4
8. Normative References . . . . .	5
Authors' Addresses . . . . .	5

## 1. Introduction

DetNet is chartered to provide deterministic forwarding over Layer 3. Deterministic forwarding means packet forwarding with bounded latency, loss and delay variation [draft-ietf-detnet-problem-statement]. In current DetNet's discussion, low packet loss is mainly achieved through PREOF (Packet Replication, Elimination, and Ordering Functions) [draft-ietf-detnet-architecture]. This document focuses on bounded latency.

Common IP/MPLS forwarding has long tail effect that couldn't guarantee bounded latency. DetNet has to have an approach to identify DetNet flows, and divert them into a DetNet forwarding plane in which some schemes are adapted to guarantee bounded latency. There are several schemes are proposed for bounded latency forwarding such as dedicated tunnel, light load with per-flow per-hop shaping, Time Aware Shaping[IEEE802.1Qbv], Cyclic Queuing and Forwarding[IEEE802.1Qch], Scalable Deterministic Forwarding[draft-qiange-detnet-large-scale-detnet], and SR based bounded latency[draft-chen-detnet-sr-based-bounded-latency]. This document is not going to compare and analyze these schemes, but to propose some factors (Layer 3 specific) worth to be considered when selecting Layer 3 bounded latency forwarding scheme.

## 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.

## 1.2. Terminology & Abbreviations

This document uses the terminology defined in [draft-ietf-detnet-architecture].

TSN: Time Sensitive Network

## 2. Tolerance of Time Deviation

One of DetNet's objectives is to stitch TSN domains together as shown in Figure 1. We know that devices inside a TSN domain are time-synchronized, and most of TSN forwarding schemes rely on precise time synchronization. However two TSN domains have a high probability of being asynchronous, while DetNet needs to connect them together and provide end-to-end deterministic forwarding. Therefore, it is worthy of have a DetNet forwarding plane which can keep the bounded latency even under an unsynchronized situation. Otherwise, buffer is needed to absorb the time deviation, and end-users have to bear the latency and cost increase introduced by buffer.

Moreover, there are

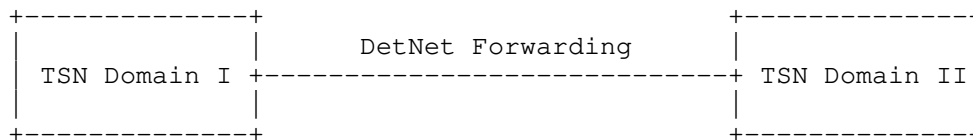


Figure 1: DetNet Scenario

## 3. Long Link Propagation Delay

In contrast to Layer 2 TSN that deployed in LAN, Layer 3 DetNet is expected to be deployed in larger scale network that has longer link propagation delay. Long link propagation delay can cause some troubles to simple cyclic forwarding schemes, like [IEEE802.1Qch]. IEEE 802.1 Qch works on the basis of time synchronization, and one hop forwarding (include packet sending, packet transmission on link, and packet receiving three operations) is required to be finished within one cycle as shown in Figure 2. Long links whose latency exceed the cycle time will make IEEE 802.1 Qch doesn't work. One possible solution is to set cycle time to be a bigger value, in order to absorb long link propagation delay. However, this solution will lead to larger jitter. The reason is that cyclic forwarding schemes only try to ensure the packet arrives at a node within a certain cycle, and packet's arrival time may vary within that cycle.

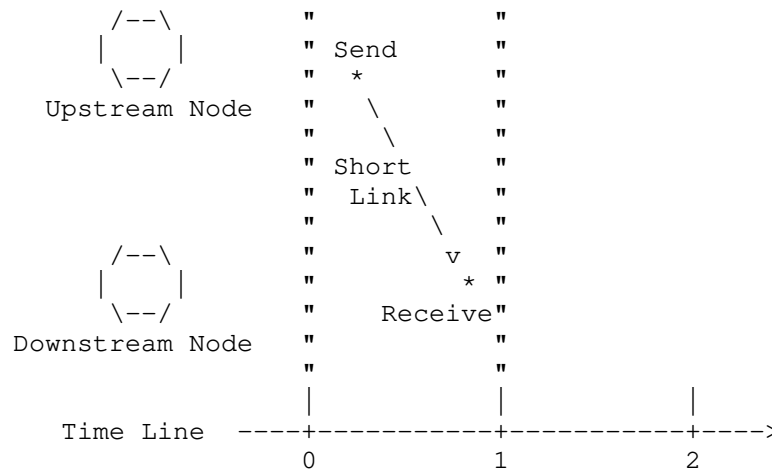


Figure 2: One Hop Forwarding in Qch

[Jitter on link is TBD]

## 4. Massive Dynamic Flows

Considering the features of TSN applications we can speculate that the number of TSN flows will not dramatically change with time. While DetNet targets at larger-scale deployment. There are more time-sensitive applications such as VR communication, they may require establishment or tear-down of the DetNet connections frequently. Meanwhile, layer 3 device may serve millions of traffic flows simultaneously. Hence those schemes that need complex calculations may not be applicable in DetNet. More importantly, forwarding schemes need to avoid impact on those in-transit flows when new flows are added (or old flows are removed).

## 5. IANA Considerations

This document makes no request of IANA.

## 6. Security Considerations

This document will not introduce new security problems.

## 7. Acknowledgements

The Authors would like to thank David Black for his review, suggestion and comments to this document.

## 8. Normative References

- [draft-chen-detnet-sr-based-bounded-latency]  
"SR based Bounded Latency",  
<[https://datatracker.ietf.org/doc/  
draft-chen-detnet-sr-based-bounded-latency/](https://datatracker.ietf.org/doc/draft-chen-detnet-sr-based-bounded-latency/)>.
- [draft-ietf-detnet-architecture]  
"DetNet Architecture", <[https://datatracker.ietf.org/doc/  
draft-ietf-detnet-architecture/](https://datatracker.ietf.org/doc/draft-ietf-detnet-architecture/)>.
- [draft-ietf-detnet-problem-statement]  
"DetNet Problem Statement",  
<[https://datatracker.ietf.org/doc/  
draft-ietf-detnet-problem-statement/](https://datatracker.ietf.org/doc/draft-ietf-detnet-problem-statement/)>.
- [draft-qiang-detnet-large-scale-detnet]  
"Large-Scale Deterministic Network",  
<[https://datatracker.ietf.org/doc/  
draft-qiang-detnet-large-scale-detnet/](https://datatracker.ietf.org/doc/draft-qiang-detnet-large-scale-detnet/)>.
- [IEEE802.1Qbv]  
"Enhancements for Scheduled Traffic", 2016.
- [IEEE802.1Qch]  
"Cyclic Queuing and Forwarding", 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

## Authors' Addresses

Liang Geng  
China Mobile  
Beijing  
China

Email: [gengliang@chinamobile.com](mailto:gengliang@chinamobile.com)

Li Qiang  
Huawei Technologies  
Huawei Campus, No. 156 Beiqing Rd.  
Beijing 100095  
China

Email: qiangli3@huawei.com

Toerless Eckert  
Huawei USA - Futurewei Technologies Inc.  
2330 Central Expy  
Santa Clara 95050  
USA

Email: tte+ietf@cs.fau.de

DetNet  
Internet-Draft  
Intended status: Standards Track  
Expires: September 10, 2019

N. Finn  
Huawei  
P. Thubert  
Cisco  
B. Varga  
J. Farkas  
Ericsson  
March 9, 2019

Deterministic Networking Architecture  
draft-ietf-detnet-architecture-12

Abstract

This document provides the overall architecture for Deterministic Networking (DetNet), which 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. Techniques used include: 1) reserving data plane resources for individual (or aggregated) DetNet flows in some or all of the intermediate nodes along the path of the flow; 2) providing explicit routes for DetNet flows that do not immediately change with the network topology; and 3) 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 operates at the IP layer and delivers service over lower layer technologies such as MPLS and IEEE 802.1 Time-Sensitive Networking (TSN).

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 September 10, 2019.

## Copyright Notice

Copyright (c) 2019 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 . . . . .	3
2. Terminology . . . . .	4
2.1. Terms used in this document . . . . .	4
2.2. IEEE 802.1 TSN to DetNet dictionary . . . . .	7
3. Providing the DetNet Quality of Service . . . . .	7
3.1. Primary goals defining the DetNet QoS . . . . .	8
3.2. Mechanisms to achieve DetNet QoS . . . . .	10
3.2.1. Resource allocation . . . . .	10
3.2.1.1. Eliminate contention loss . . . . .	10
3.2.1.2. Jitter Reduction . . . . .	11
3.2.2. Service Protection . . . . .	11
3.2.2.1. In-Order Delivery . . . . .	12
3.2.2.2. Packet Replication and Elimination . . . . .	12
3.2.2.3. Packet encoding for service protection . . . . .	14
3.2.3. Explicit routes . . . . .	14
3.3. Secondary goals for DetNet . . . . .	15
3.3.1. Coexistence with normal traffic . . . . .	15
3.3.2. Fault Mitigation . . . . .	16
4. DetNet Architecture . . . . .	17
4.1. DetNet stack model . . . . .	17
4.1.1. Representative Protocol Stack Model . . . . .	17
4.1.2. DetNet Data Plane Overview . . . . .	20
4.1.3. Network reference model . . . . .	22
4.2. DetNet systems . . . . .	23
4.2.1. End system . . . . .	23
4.2.2. DetNet edge, relay, and transit nodes . . . . .	24
4.3. DetNet flows . . . . .	25
4.3.1. DetNet flow types . . . . .	25
4.3.2. Source transmission behavior . . . . .	25
4.3.3. Incomplete Networks . . . . .	27
4.4. Traffic Engineering for DetNet . . . . .	27

4.4.1. The Application Plane . . . . .	28
4.4.2. The Controller Plane . . . . .	28
4.4.3. The Network Plane . . . . .	29
4.5. Queuing, Shaping, Scheduling, and Preemption . . . . .	30
4.6. Service instance . . . . .	31
4.7. Flow identification at technology borders . . . . .	32
4.7.1. Exporting flow identification . . . . .	32
4.7.2. Flow attribute mapping between layers . . . . .	34
4.7.3. Flow-ID mapping examples . . . . .	35
4.8. Advertising resources, capabilities and adjacencies . . . . .	36
4.9. Scaling to larger networks . . . . .	37
4.10. Compatibility with Layer-2 . . . . .	37
5. Security Considerations . . . . .	37
6. Privacy Considerations . . . . .	39
7. IANA Considerations . . . . .	39
8. Acknowledgements . . . . .	39
9. Informative References . . . . .	40
Authors' Addresses . . . . .	44

## 1. Introduction

This document provides the overall architecture for Deterministic Networking (DetNet), which provides a capability for the delivery of data flows with extremely low packet loss rates and bounded end-to-end delivery latency. DetNet is for networks that are under a single administrative control or within a closed group of administrative control; these include campus-wide networks and private WANs. DetNet is not for large groups of domains such as the Internet.

DetNet operates at the IP layer and delivers service over lower layer technologies such as MPLS and IEEE 802.1 Time-Sensitive Networking (TSN). DetNet accomplishes these goals by dedicating network resources such as link bandwidth and buffer space to DetNet flows and/or classes of DetNet flows, and by replicating packets along multiple paths. Unused reserved resources are available to non-DetNet packets as long as all guarantees are fulfilled.

The Deterministic Networking Problem Statement

[I-D.ietf-detnet-problem-statement] introduces Deterministic Networking, and Deterministic Networking Use Cases

[I-D.ietf-detnet-use-cases] summarizes the need for it. See [I-D.ietf-detnet-dp-sol-mpls] and [I-D.ietf-detnet-dp-sol-ip] for specific techniques that can be used to identify DetNet flows and assign them to specific paths through a network.

A goal of DetNet is a converged network in all respects including the convergence of sensitive non-IP networks onto a common network infrastructure. The presence of DetNet flows does not preclude non-



DetNet flows, and the benefits offered DetNet flows should not, except in extreme cases, prevent existing Quality of Service (QoS) mechanisms from operating in a normal fashion, subject to the bandwidth required for the DetNet flows. A single source-destination pair can trade both DetNet and non-DetNet flows. End systems and applications need not instantiate special interfaces for DetNet flows. Networks are not restricted to certain topologies; connectivity is not restricted. Any application that generates a data flow that can be usefully characterized as having a maximum bandwidth should be able to take advantage of DetNet, as long as the necessary resources can be reserved. Reservations can be made by the application itself, via network management, by an application's controller, or by other means, e.g., a dynamic control plane (e.g., [RFC2205]). QoS requirements of DetNet flows can be met if all network nodes in a DetNet domain implement DetNet capabilities. DetNet nodes can be interconnected with different sub-network technologies (Section 4.1.2), where the nodes of the subnet are not DetNet aware (Section 4.1.3).

Many applications that are intended to be served by Deterministic Networking require the ability to synchronize the clocks in end systems to a sub-microsecond accuracy. Some of the queue control techniques defined in Section 4.5 also require time synchronization among network nodes. The means used to achieve time synchronization are not addressed in this document. DetNet can accommodate various time synchronization techniques and profiles that are defined elsewhere to address the needs of different market segments.

## 2. Terminology

### 2.1. Terms used in this document

The following terms are used in the context of DetNet in this document:

#### allocation

Resources are dedicated to support a DetNet flow. Depending on an implementation, the resource may be reused by non-DetNet flows when it is not used by the DetNet flow.

#### App-flow

The payload (data) carried over a DetNet service.

#### DetNet compound flow and DetNet member flow

A DetNet compound flow is a DetNet flow that has been separated into multiple duplicate DetNet member flows for service protection at the DetNet service sub-layer. Member flows are merged back into a single DetNet compound flow such

that there are no duplicate packets. "Compound" and "member" are strictly relative to each other, not absolutes; a DetNet compound flow comprising multiple DetNet member flows can, in turn, be a member of a higher-order compound.

DetNet destination

An end system capable of terminating a DetNet flow.

DetNet domain

The portion of a network that is DetNet aware. It includes end systems and DetNet nodes.

DetNet edge node

An instance of a DetNet relay node that acts as a source and/or destination at the DetNet service sub-layer. For example, it can include a DetNet service sub-layer proxy function for DetNet service protection (e.g., the addition or removal of packet sequencing information) for one or more end systems, or starts or terminates resource allocation at the DetNet forwarding sub-layer, or aggregates DetNet services into new DetNet flows. It is analogous to a Label Edge Router (LER) or a Provider Edge (PE) router.

DetNet flow

A DetNet flow is a sequence of packets which conform uniquely to a flow identifier, and to which the DetNet service is to be provided. It includes any DetNet headers added to support the DetNet service and forwarding sub-layers.

DetNet forwarding sub-layer

DetNet functionality is divided into two sub-layers. One of them is the DetNet forwarding sub-layer, which optionally provides resource allocation for DetNet flows over paths provided by the underlying network.

DetNet intermediate node

A DetNet relay node or DetNet transit node.

DetNet node

A DetNet edge node, a DetNet relay node, or a DetNet transit node.

DetNet relay node

A DetNet node including a service sub-layer function that interconnects different DetNet forwarding sub-layer paths to provide service protection. A DetNet relay node participates in the DetNet service sub-layer. It typically incorporates

DetNet forwarding sub-layer functions as well, in which case it is collocated with a transit node.

DetNet service sub-layer

DetNet functionality is divided into two sub-layers. One of them is the DetNet service sub-layer, at which a DetNet service, e.g., service protection is provided.

DetNet service proxy

Maps between App-flows and DetNet flows.

DetNet source

An end system capable of originating a DetNet flow.

DetNet system

A DetNet aware end system, transit node, or relay node. "DetNet" may be omitted in some text.

DetNet transit node

A DetNet node operating at the DetNet forwarding sub-layer, that utilizes link layer and/or network layer switching across multiple links and/or sub-networks to provide paths for DetNet service sub-layer functions. Typically provides resource allocation over those paths. An MPLS LSR is an example of a DetNet transit node.

DetNet-UNI

User-to-Network Interface with DetNet specific functionalities. It is a packet-based reference point and may provide multiple functions like encapsulation, status, synchronization, etc.

end system

Commonly called a "host" in IETF documents, and an "end station" in IEEE 802 documents. End systems of interest to this document are either sources or destinations of DetNet flows. An end system may or may not be DetNet forwarding sub-layer aware or DetNet service sub-layer aware.

link

A connection between two DetNet nodes. It may be composed of a physical link or a sub-network technology that can provide appropriate traffic delivery for DetNet flows.

PEF

A Packet Elimination Function (PEF) eliminates duplicate copies of packets to prevent excess packets flooding the network or duplicate packets being sent out of the DetNet

domain. PEF can be implemented by a DetNet edge node, a DetNet relay node, or an end system.

PRF      A Packet Replication Function (PRF) replicates DetNet flow packets and forwards them to one or more next hops in the DetNet domain. The number of packet copies sent to the next hops is a DetNet flow specific parameter at the point of replication. PRF can be implemented by a DetNet edge node, a DetNet relay node, or an end system.

PREOF    Collective name for Packet Replication, Elimination, and Ordering Functions.

POF      A Packet Ordering Function (POF) re-orders packets within a DetNet flow that are received out of order. This function can be implemented by a DetNet edge node, a DetNet relay node, or an end system.

reservation

The set of resources allocated between a source and one or more destinations through DetNet nodes and subnets associated with a DetNet flow, to provide the provisioned DetNet service.

## 2.2. IEEE 802.1 TSN to DetNet dictionary

This section also serves as a dictionary for translating from the terms used by the Time-Sensitive Networking (TSN) Task Group [IEEE802.1TSNTG] of the IEEE 802.1 WG to those of the DetNet WG.

Listener

The IEEE 802.1 term for a destination of a DetNet flow.

relay system

The IEEE 802.1 term for a DetNet intermediate node.

Stream

The IEEE 802.1 term for a DetNet flow.

Talker

The IEEE 802.1 term for the source of a DetNet flow.

## 3. Providing the DetNet Quality of Service

### 3.1. Primary goals defining the DetNet QoS

The DetNet Quality of Service can be expressed in terms of:

- o Minimum and maximum end-to-end latency from source to destination; timely delivery, and bounded jitter (packet delay variation) derived from these constraints.
- o Packet loss ratio, under various assumptions as to the operational states of the nodes and links.
- o 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.

It is a distinction of DetNet that it is concerned solely with worst-case values for the end-to-end latency, jitter, and misordering. Average, mean, or typical values are of little interest, because they do not affect the ability of a real-time system to perform its tasks. In general, a trivial priority-based queuing scheme will give better average latency to a data flow than DetNet; however, it may not be a suitable option for DetNet because of its worst-case latency.

Three techniques are used by DetNet to provide these qualities of service:

- o Resource allocation (Section 3.2.1).
- o Service protection (Section 3.2.2).
- o Explicit routes (Section 3.2.3).

Resource allocation operates by assigning resources, e.g., buffer space or link bandwidth, to a DetNet flow (or flow aggregate) along its path. Resource allocation greatly reduces, or even eliminates entirely, packet loss due to output packet contention within the network, but it can only be supplied to a DetNet flow that is limited at the source to a maximum packet size and transmission rate. As DetNet flows are assumed to be rate-limited and DetNet is designed to provide sufficient allocated resources (including provisioned capacity), the use of transport layer congestion control [RFC2914] for App-flows is not required; however, if resources are allocated appropriately, use of congestion control should not impact transmission negatively.

Resource allocation addresses two of the DetNet QoS requirements: latency and packet loss. Given that DetNet nodes have a finite amount of buffer space, resource allocation necessarily results in a

maximum end-to-end latency. It also addresses contention related packet loss.

Other important contribution to packet loss are random media errors and equipment failures. Service protection is the name for the mechanisms used by DetNet to address these losses. The mechanisms employed are constrained by the requirement to meet the users' latency requirements. Packet replication and elimination (Section 3.2.2) and packet encoding (Section 3.2.2.3) are described in this document to provide service protection; others may be found. For instance, packet encoding can be used to provide service protection against random media errors, packet replication and elimination can be used to provide service protection against equipment failures. This mechanism distributes the contents of DetNet flows over multiple paths in time and/or space, so that the loss of some of the paths does need not cause the loss of any packets.

The paths are typically (but not necessarily) explicit routes, so that they do not normally suffer temporary interruptions caused by the convergence of routing or bridging protocols.

These three techniques can be applied independently, giving eight possible combinations, including none (no DetNet), although some combinations are of wider utility than others. This separation keeps the protocol stack coherent and maximizes interoperability with existing and developing standards in this (IETF) and other Standards Development Organizations. Some examples of typical expected combinations:

- o Explicit routes plus service protection are exactly the techniques employed by seamless redundancy mechanisms applied on a ring topology as described, e.g., in [IEC62439-3-2016]. In this example, explicit routes are achieved by limiting the physical topology of the network to a ring. Sequentialization, replication, and duplicate elimination are facilitated by packet tags added at the front or the end of Ethernet frames. [RFC8227] provides another example in the context of MPLS.
- o Resource allocation alone was originally offered by IEEE 802.1 Audio Video bridging [IEEE802.1BA]. As long as the network suffers no failures, packet loss due to output packet contention can be eliminated through the use of a reservation protocol (e.g., Multiple Stream Registration Protocol [IEEE802.1Q-2018]), shapers in every bridge, and proper dimensioning.
- o Using all three together gives maximum protection.

There are, of course, simpler methods available (and employed, today) to achieve levels of latency and packet loss that are satisfactory for many applications. Prioritization and over-provisioning is one such technique. However, these methods generally work best in the absence of any significant amount of non-critical traffic in the network (if, indeed, such traffic is supported at all), or work only if the critical traffic constitutes only a small portion of the network's theoretical capacity, or work only if all systems are functioning properly, or in the absence of actions by end systems that disrupt the network's operations.

There are any number of methods in use, defined, or in progress for accomplishing each of the above techniques. It is expected that this DetNet Architecture will assist various vendors, users, and/or "vertical" Standards Development Organizations (dedicated to a single industry) to make selections among the available means of implementing DetNet networks.

### 3.2. Mechanisms to achieve DetNet QoS

#### 3.2.1. Resource allocation

##### 3.2.1.1. Eliminate contention loss

The primary means by which DetNet achieves its QoS assurances is to reduce, or even completely eliminate packet loss due to output packet contention within a DetNet node as a cause of packet loss. This can be achieved only by the provision of sufficient buffer storage at each node through the network to ensure that no packets are dropped due to a lack of buffer storage. Note that App-flows are generally not expected to be responsive to implicit [RFC2914] or explicit congestion notification [RFC3168].

Ensuring adequate buffering requires, in turn, that the source, and every DetNet node along the path to the destination (or nearly every node, see Section 4.3.3) be careful to regulate its output to not exceed the data rate for any DetNet flow, except for brief periods when making up for interfering traffic. Any packet sent ahead of its time potentially adds to the number of buffers required by the next hop DetNet node and may thus exceed the resources allocated for a particular DetNet flow. Furthermore, rate limiting, e.g., using traffic policing and shaping functions, e.g., [RFC2475], at the ingress of the DetNet domain must be applied. This is needed for meeting the requirements of DetNet flows as well as for protecting non-DetNet traffic from potentially misbehaving DetNet traffic sources. Note that large buffers have some issues, see, e.g., [BUFFERBLOAT].

The low-level mechanisms described in Section 4.5 provide the necessary regulation of transmissions by an end system or DetNet node to provide resource allocation. The allocation of the bandwidth and buffers for a DetNet flow requires provisioning. A DetNet node may have other resources requiring allocation and/or scheduling, that might otherwise be over-subscribed and trigger the rejection of a reservation.

#### 3.2.1.2. Jitter Reduction

A core objective of DetNet is to enable the convergence of sensitive non-IP networks onto a common network infrastructure. This requires the accurate emulation of currently deployed mission-specific networks, which for example rely on point-to-point analog (e.g., 4-20mA modulation) and serial-digital cables (or buses) for highly reliable, synchronized and jitter-free communications. While the latency of analog transmissions is basically the speed of light, legacy serial links are usually slow (in the order of Kbps) compared to, say, Gigabit Ethernet, and some latency is usually acceptable. What is not acceptable is the introduction of excessive jitter, which may, for instance, affect the stability of control systems.

Applications that are designed to operate on serial links usually do not provide services to recover the jitter, because jitter simply does not exist there. DetNet flows are generally expected to be delivered in-order and the precise time of reception influences the processes. In order to converge such existing applications, there is a desire to emulate all properties of the serial cable, such as clock transportation, perfect flow isolation and fixed latency. While minimal jitter (in the form of specifying minimum, as well as maximum, end-to-end latency) is supported by DetNet, there are practical limitations on packet-based networks in this regard. In general, users are encouraged to use a combination of:

- o Sub-microsecond time synchronization among all source and destination end systems, and
- o Time-of-execution fields in the application packets.

Jitter reduction is provided by the mechanisms described in Section 4.5 that also provide resource allocation.

#### 3.2.2. Service Protection

Service protection aims to mitigate or eliminate packet loss due to equipment failures, including random media and/or memory faults. These types of packet loss can be greatly reduced by spreading the data over multiple disjoint forwarding paths. Various service



protection methods are described in [RFC6372], e.g., 1+1 linear protection. This section describes the functional details of an additional method in Section 3.2.2.2, which can be implemented as described in Section 3.2.2.3 or as specified in [I-D.ietf-detnet-dp-sol-mpls] in order to provide 1+n hitless protection. The appropriate service protection mechanism depends on the scenario and the requirements.

#### 3.2.2.1. In-Order Delivery

Out-of-order packet delivery can be a side effect of service protection. Packets delivered out-of-order impact the amount of buffering needed at the destination to properly process the received data. Such packets also influence the jitter of a flow. The DetNet service includes maximum allowed misordering as a constraint. Zero misordering would be a valid service constraint to reflect that the end system(s) of the flow cannot tolerate any out-of-order delivery. DetNet Packet Ordering Functionality (POF) (Section 3.2.2.2) can be used to provide in-order delivery.

#### 3.2.2.2. Packet Replication and Elimination

This section describes a service protection method that sends copies of the same packets over multiple paths.

The DetNet service sub-layer includes the packet replication (PRF), the packet elimination (PEF), and the packet ordering functionality (POF) for use in DetNet edge, relay node, and end system packet processing. These functions can be enabled in a DetNet edge node, relay node or end system. The collective name for all three functions is Packet Replication, Elimination, and Ordering Functions (PREOF). The packet replication and elimination service protection method altogether involves four capabilities:

- o Providing sequencing information to the packets of a DetNet compound flow. This may be done by adding a sequence number or time stamp as part of DetNet, or may be inherent in the packet, e.g., in a higher layer protocol, or associated to other physical properties such as the precise time (and radio channel) of reception of the packet. This is typically done once, at or near the source.
- o The Packet Replication Function (PRF) replicates these packets into multiple DetNet member flows and typically sends them along multiple different paths to the destination(s), e.g., over the explicit routes of Section 3.2.3. The location within a DetNet node, and the mechanism used for the PRF is left open for implementations.

- o The Packet Elimination Function (PEF) eliminates duplicate packets of a DetNet flow based on the sequencing information and a history of received packets. The output of the PEF is always a single packet. This may be done at any DetNet node along the path to save network resources further downstream, in particular if multiple Replication points exist. But the most common case is to perform this operation at the very edge of the DetNet network, preferably in or near the receiver. The location within a DetNet node, and mechanism used for the PEF is left open for implementations.
- o The Packet Ordering Function (POF) uses the sequencing information to re-order a DetNet flow's packets that are received out of order.

The order in which a DetNet node applies PEF, POF, and PRF to a DetNet flow is left open for implementations.

Some service protection mechanisms rely on switching from one flow to another when a failure of a flow is detected. Contrarily, packet replication and elimination combines the DetNet member flows sent along multiple different paths, and performs a packet-by-packet selection of which to discard, e.g., based on sequencing information.

In the simplest case, this amounts to replicating each packet in a source that has two interfaces, and conveying them through the network, along separate (Shared Risk Link Group (SRLG) disjoint) paths, to the similarly dual-homed destinations, that discard the extras. This ensures that one path remains, even if some DetNet intermediate node fails. The sequencing information can also be used for loss detection and for re-ordering.

DetNet relay nodes in the network can provide replication and elimination facilities at various points in the network, so that multiple failures can be accommodated.

This is shown in Figure 1, where the two relay nodes each replicate (R) the DetNet flow on input, sending the DetNet member flows to both the other relay node and to the end system, and eliminate duplicates (E) on the output interface to the right-hand end system. Any one link in the network can fail, and the DetNet compound flow can still get through. Furthermore, two links can fail, as long as they are in different segments of the network.

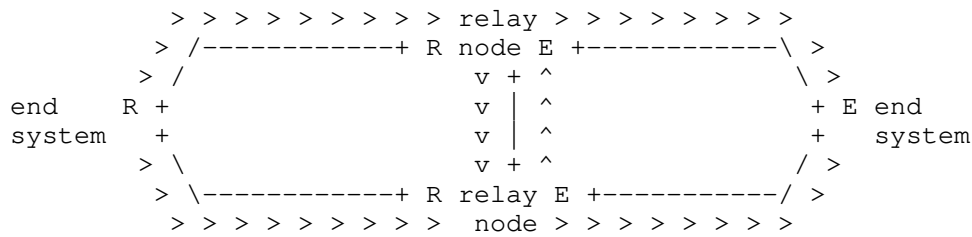


Figure 1: Packet replication and elimination

Packet replication and elimination does not react to and correct failures; it is entirely passive. Thus, intermittent failures, mistakenly created packet filters, or misrouted data is handled just the same as the equipment failures that are handled by typical routing and bridging protocols.

If member flows that take different-length paths through the network are combined, a merge point may require extra buffering to equalize the delays over the different paths. This equalization ensures that the resultant compound flow will not exceed its contracted bandwidth even after one or the other of the paths is restored after a failure. The extra buffering can be also used to provide in-order delivery.

### 3.2.2.3. Packet encoding for service protection

There are methods for using multiple paths to provide service protection that involve encoding the information in a packet belonging to a DetNet flow into multiple transmission units, combining information from multiple packets into any given transmission unit. Such techniques, also known as "network coding", can be used as a DetNet service protection technique.

### 3.2.3. Explicit routes

In networks controlled by typical dynamic control protocols such as IS-IS or OSPF, a network topology event in one part of the network can impact, at least briefly, the delivery of data in parts of the network remote from the failure or recovery event. Even the use of redundant paths through a network, e.g., as defined by [RFC6372] do not eliminate the chances of packet loss. Furthermore, out-of-order packet delivery can be a side effect of route changes.

Many real-time networks rely on physical rings of two-port devices, with a relatively simple ring control protocol. This supports redundant paths for service protection with a minimum of wiring. As an additional benefit, ring topologies can often utilize different topology management protocols than those used for a mesh network,

with a consequent reduction in the response time to topology changes. Of course, this comes at some cost in terms of increased hop count, and thus latency, for the typical path.

In order to get the advantages of low hop count and still ensure against even very brief losses of connectivity, DetNet employs explicit routes, where the path taken by a given DetNet flow does not change, at least immediately, and likely not at all, in response to network topology events. Service protection (Section 3.2.2 or Section 3.2.2.3) over explicit routes provides a high likelihood of continuous connectivity. Explicit routes can be established in various ways, e.g., with RSVP-TE [RFC3209], with Segment Routing (SR) [RFC8402], via a Software Defined Networking approach [RFC8453], with IS-IS [RFC7813], etc. Explicit routes are typically used in MPLS TE LSPs.

Out-of-order packet delivery can be a side effect of distributing a single flow over multiple paths, especially when there is a change from one path to another when combining the flow. This is irrespective of the distribution method used, and also applies to service protection over explicit routes. As described in Section 3.2.2.1, out-of-order packets influence the jitter of a flow and impact the amount of buffering needed to process the data; therefore, DetNet service includes maximum allowed misordering as a constraint. The use of explicit routes helps to provide in-order delivery because there is no immediate route change with the network topology, but the changes are plannable as they are between the different explicit routes.

### 3.3. Secondary goals for DetNet

Many applications require DetNet to provide additional services, including coexistence with other QoS mechanisms Section 3.3.1 and protection against misbehaving transmitters Section 3.3.2.

#### 3.3.1. Coexistence with normal traffic

A DetNet network supports the dedication of a high proportion of the network bandwidth to DetNet flows. But, no matter how much is dedicated for DetNet flows, it is a goal of DetNet to coexist with existing Class of Service schemes (e.g., DiffServ). It is also important that non-DetNet traffic not disrupt the DetNet flow, of course (see Section 3.3.2 and Section 5). For these reasons:

- o Bandwidth (transmission opportunities) not utilized by a DetNet flow is available to non-DetNet packets (though not to other DetNet flows).

- o DetNet flows can be shaped or scheduled, in order to ensure that the highest-priority non-DetNet packet is also ensured a worst-case latency.
- o When transmission opportunities for DetNet flows are scheduled in detail, then the algorithm constructing the schedule should leave sufficient opportunities for non-DetNet packets to satisfy the needs of the users of the network. Detailed scheduling can also permit the time-shared use of buffer resources by different DetNet flows.

Starvation of non-DetNet traffic must be avoided, e.g., by traffic policing and shaping functions (e.g., [RFC2475]). Thus, the net effect of the presence of DetNet flows in a network on the non-DetNet flows is primarily a reduction in the available bandwidth.

### 3.3.2. Fault Mitigation

Robust real-time systems require reducing the number of possible failures. Filters and policers should be used in a DetNet network to detect if DetNet packets are received on the wrong interface, or at the wrong time, or in too great a volume. Furthermore, filters and policers can take actions to discard the offending packets or flows, or trigger shutting down the offending flow or the offending interface.

It is also essential that filters and service remarking be employed at the network edge to prevent non-DetNet packets from being mistaken for DetNet packets, and thus impinging on the resources allocated to DetNet packets. In particular, sending DetNet traffic into networks that have not been provisioned in advance to handle that DetNet traffic has to be treated as a fault. The use of egress traffic filters, or equivalent mechanisms, to prevent this from happening are strongly recommended at the edges of a DetNet networks and DetNet supporting networks. In this context, the term 'provisioned' has a broad meaning, e.g., provisioning could be performed via an administrative decision that the downstream network has the available capacity to carry the DetNet traffic that is being sent into it.

Note that the sending of App-flows that do not use transport layer congestion control per [RFC2914] into a network that is not provisioned to handle such DetNet traffic has to be treated as a fault and prevented. PRF generated DetNet member flows also need to be treated as not using transport layer congestion control even if the original App-flow supports transport layer congestion control because PREOF can remove congestion indications at the PEF and thereby hide such indications (e.g., drops, ECN markings, increased latency) from end systems.

The mechanisms to support these requirements are both data plane and implementation specific. Data plane specific solutions will be specified in the relevant data plane solution document. There also exist techniques, at present and/or in various stages of standardization, that can support these fault mitigation tasks that deliver a high probability that misbehaving systems will have zero impact on well-behaved DetNet flows, except of course, for the receiving interface(s) immediately downstream of the misbehaving device. Examples of such techniques include traffic policing and shaping functions (e.g., [RFC2475]) and separating flows into per-flow rate-limited queues.

#### 4. DetNet Architecture

##### 4.1. DetNet stack model

DetNet functionality (Section 3) is implemented in two adjacent sub-layers in the protocol stack: the DetNet service sub-layer and the DetNet forwarding sub-layer. The DetNet service sub-layer provides DetNet service, e.g., service protection, to higher layers in the protocol stack and applications. The DetNet forwarding sub-layer supports DetNet service in the underlying network, e.g., by providing explicit routes and resource allocation to DetNet flows.

##### 4.1.1. Representative Protocol Stack Model

Figure 2 illustrates a conceptual DetNet data plane layering model. One may compare it to that in [IEEE802.1CB], Annex C.

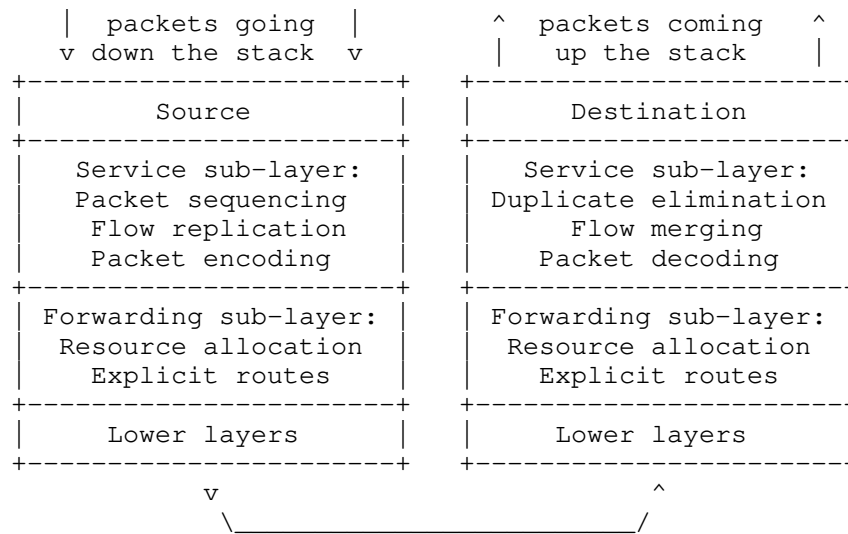


Figure 2: DetNet data plane protocol stack

Not all sub-layers are required for any given application, or even for any given network. The functionality shown in Figure 2 is:

#### Application

Shown as "source" and "destination" in the diagram.

#### Packet sequencing

As part of DetNet service protection, supplies the sequence number for packet replication and elimination (Section 3.2.2), thus peers with Duplicate elimination. This sub-layer is not needed if a higher layer protocol is expected to perform any packet sequencing and duplicate elimination required by the DetNet flow replication.

#### Duplicate elimination

As part of the DetNet service sub-layer, based on the sequenced number supplied by its peer, packet sequencing, Duplicate elimination discards any duplicate packets generated by DetNet flow replication. It can operate on member flows, compound flows, or both. The replication may also be inferred from other information such as the precise time of reception in a scheduled network. The duplicate elimination sub-layer may also perform resequencing of packets to restore packet order in a flow that was disrupted by the loss of packets on one or another of the multiple paths taken.

#### Flow replication

As part of DetNet service protection, packets that belong to a DetNet compound flow are replicated into two or more DetNet member flows. This function is separate from packet sequencing. Flow replication can be an explicit replication and remarking of packets, or can be performed by, for example, techniques similar to ordinary multicast replication, albeit with resource allocation implications. Peers with DetNet flow merging.

#### Flow merging

As part of DetNet service protection, merges DetNet member flows together for packets coming up the stack belonging to a specific DetNet compound flow. Peers with DetNet flow replication. DetNet flow merging, together with packet sequencing, duplicate elimination, and DetNet flow replication perform packet replication and elimination (Section 3.2.2).

#### Packet encoding

As part of DetNet service protection, as an alternative to packet sequencing and flow replication, packet encoding combines the information in multiple DetNet packets, perhaps from different DetNet compound flows, and transmits that information in packets on different DetNet member Flows. Peers with Packet decoding.

#### Packet decoding

As part of DetNet service protection, as an alternative to flow merging and duplicate elimination, packet decoding takes packets from different DetNet member flows, and computes from those packets the original DetNet packets from the compound flows input to packet encoding. Peers with Packet encoding.

#### Resource allocation

The DetNet forwarding sub-layer provides resource allocation. See Section 4.5. The actual queuing and shaping mechanisms are typically provided by underlying subnet. These can be closely associated with the means of providing paths for DetNet flows. The path and the resource allocation are conflated in this figure.

#### Explicit routes

The DetNet forwarding sub-layer provides mechanisms to ensure that fixed paths are provided for DetNet flows. These explicit paths avoid the impact of network convergence.



Operations, Administration, and Maintenance (OAM) leverages in-band and out-of-band signaling that validates whether the service is effectively obtained within QoS constraints. OAM is not shown in Figure 2; it may reside in any number of the layers. OAM can involve specific tagging added in the packets for tracing implementation or network configuration errors; traceability enables to find whether a packet is a replica, which DetNet relay node performed the replication, and which segment was intended for the replica. Active and hybrid OAM methods require additional bandwidth to perform fault management and performance monitoring of the DetNet domain. OAM may, for instance, generate special test probes or add OAM information into the data packet.

The packet sequencing and replication elimination functions at the source and destination ends of a DetNet compound flow may be performed either in the end system or in a DetNet relay node.

#### 4.1.2. DetNet Data Plane Overview

A "Deterministic Network" will be composed of DetNet enabled end systems, DetNet edge nodes, and DetNet relay nodes, which collectively deliver DetNet services. DetNet relay and edge nodes are interconnected via DetNet transit nodes (e.g., LSRs) which support DetNet, but are not DetNet service aware. All DetNet nodes are connected to sub-networks, where a point-to-point link is also considered as a simple sub-network. These sub-networks will provide DetNet compatible service for support of DetNet traffic. Examples of sub-network technologies include MPLS TE, IEEE 802.1 TSN and OTN. Of course, multi-layer DetNet systems may also be possible, where one DetNet appears as a sub-network, and provides service to, a higher layer DetNet system. A simple DetNet concept network is shown in Figure 3. Note that in this and following figures "Forwarding" and "Fwd" refer to the DetNet forwarding sub-layer, "Service" and "Svc" refer to the DetNet service sub-layer, which are described in detail in Section 4.1.

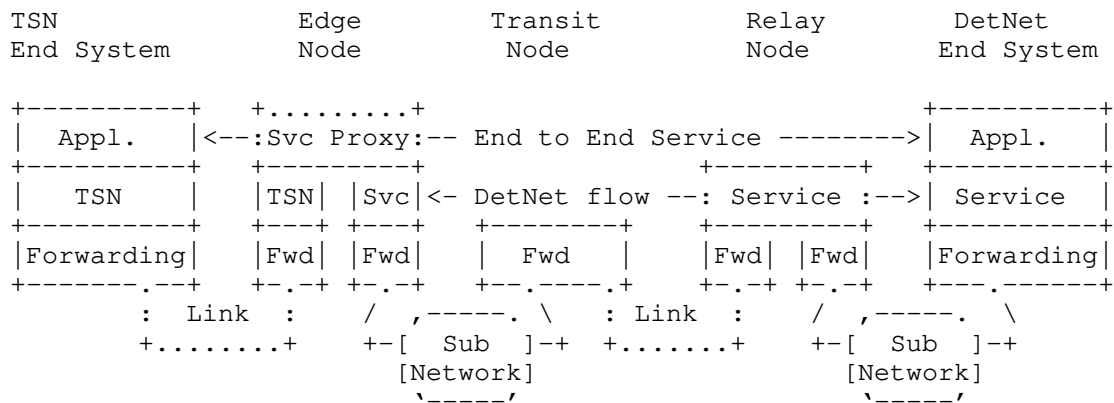


Figure 3: A Simple DetNet Enabled Network

DetNet data plane is divided into two sub-layers: the DetNet service sub-layer and the DetNet forwarding sub-layer. This helps to explore and evaluate various combinations of the data plane solutions available. Some of them are illustrated in Figure 4. This separation of DetNet sub-layers, while helpful, should not be considered as formal requirement. For example, some technologies may violate these strict sub-layers and still be able to deliver a DetNet service.

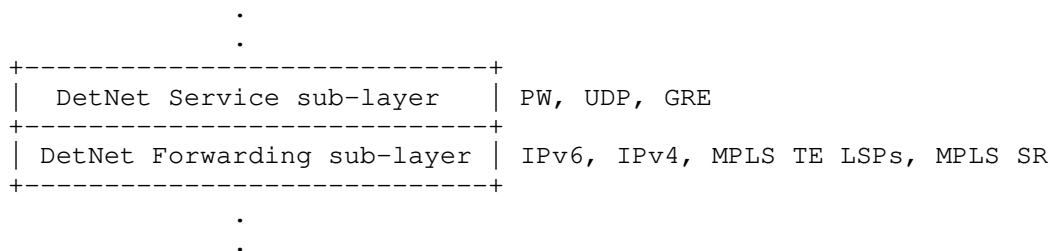


Figure 4: DetNet adaptation to data plane

In some networking scenarios, the end system initially provides a DetNet flow encapsulation, which contains all information needed by DetNet nodes (e.g., Real-time Transport Protocol (RTP) [RFC3550] based DetNet flow carried over a native UDP/IP network or PseudoWire). In other scenarios, the encapsulation formats might differ significantly.

There are many valid options to create a data plane solution for DetNet traffic by selecting a technology approach for the DetNet service sub-layer and also selecting a technology approach for the

DetNet forwarding sub-layer. There are a large number of valid combinations.

One of the most fundamental differences between different potential data plane options is the basic headers used by DetNet nodes. For example, the basic service can be delivered based on an MPLS label or an IP header. This decision impacts the basic forwarding logic for the DetNet service sub-layer. Note that in both cases, IP addresses are used to address DetNet nodes. The selected DetNet forwarding sub-layer technology also needs to be mapped to the sub-net technology used to interconnect DetNet nodes. For example, DetNet flows will need to be mapped to TSN Streams.

#### 4.1.3. Network reference model

Figure 5 shows another view of the DetNet service related reference points and main components.

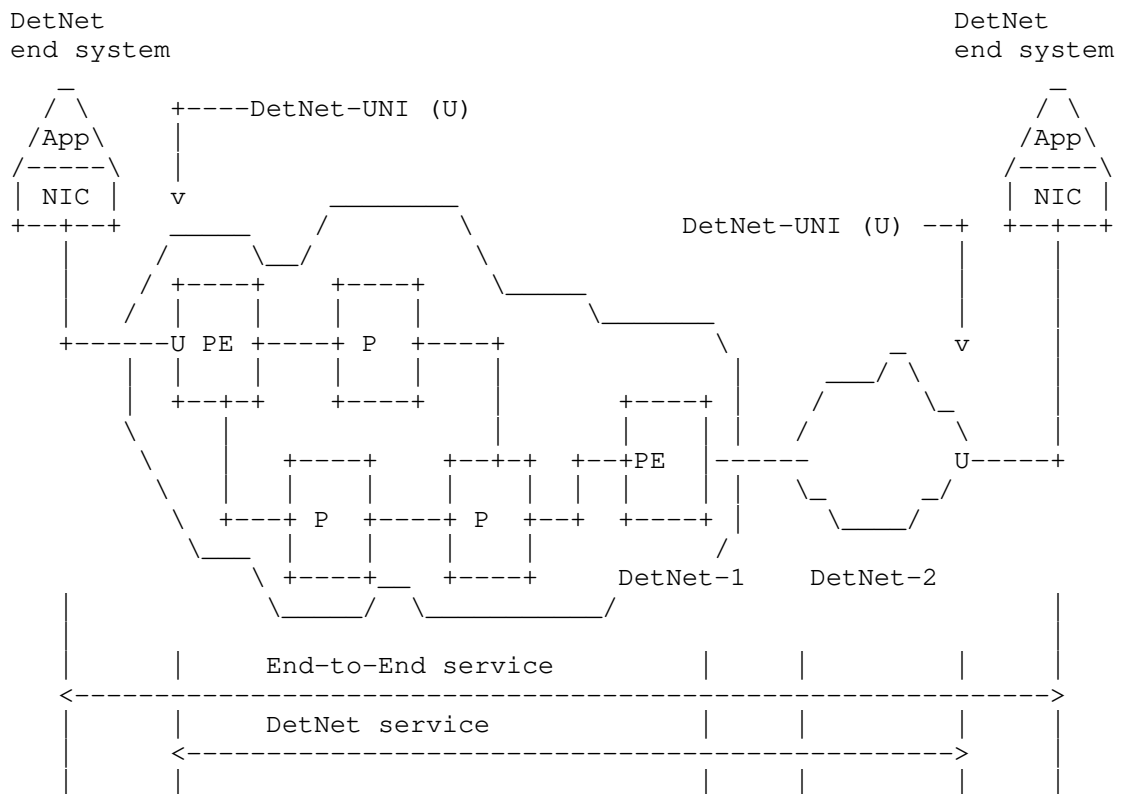


Figure 5: DetNet Service Reference Model (multi-domain)

DetNet User Network Interfaces (DetNet-UNIs) ("U" in Figure 5) are assumed in this document to be packet-based reference points and provide connectivity over the packet network. A DetNet-UNI may provide multiple functions, e.g., it may add networking technology specific encapsulation to the DetNet flows if necessary; it may provide status of the availability of the resources associated with a reservation; it may provide a synchronization service for the end system; it may carry enough signaling to place the reservation in a network without a controller, or if the controller only deals with the network but not the end systems. Internal reference points of end systems (between the application and the NIC) are more challenging from control perspective and they may have extra requirements (e.g., in-order delivery is expected in end system internal reference points, whereas it is considered optional over the DetNet-UNI).

#### 4.2. DetNet systems

##### 4.2.1. End system

The traffic characteristics of an App-flow can be CBR (constant bit rate) or VBR (variable bit rate) and can have Layer-1 or Layer-2 or Layer-3 encapsulation (e.g., TDM (time-division multiplexing), Ethernet, IP). These characteristics are considered as input for resource reservation and might be simplified to ensure determinism during packet forwarding (e.g., making reservations for the peak rate of VBR traffic, etc.).

An end system may or may not be DetNet forwarding sub-layer aware or DetNet service sub-layer aware. That is, an end system may or may not contain DetNet specific functionality. End systems with DetNet functionalities may have the same or different forwarding sub-layer as the connected DetNet domain. Categorization of end systems are shown in Figure 6.

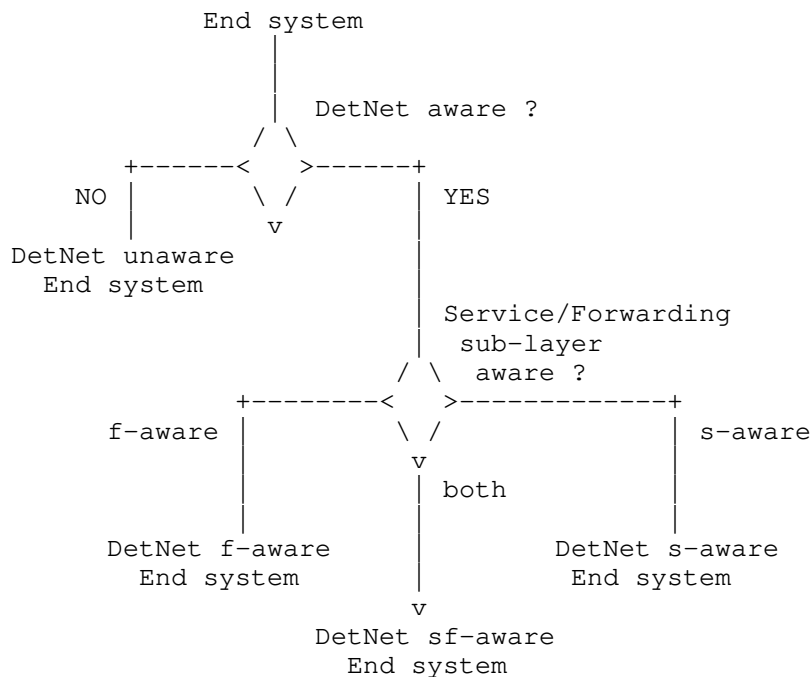


Figure 6: Categorization of end systems

Note some known use case examples for end systems:

- o DetNet unaware: The classic case requiring service proxies.
- o DetNet f-aware: A DetNet forwarding sub-layer aware system. It knows about some TSN functions (e.g., reservation), but not about service protection.
- o DetNet s-aware: A DetNet service sub-layer aware system. It supplies sequence numbers, but doesn't know about resource allocation.
- o DetNet sf-aware: A full functioning DetNet end system, it has DetNet functionalities and usually the same forwarding paradigm as the connected DetNet domain. It can be treated as an integral part of the DetNet domain.

#### 4.2.2. DetNet edge, relay, and transit nodes

As shown in Figure 3, DetNet edge nodes providing proxy service and DetNet relay nodes providing the DetNet service sub-layer are DetNet-

aware, and DetNet transit nodes need only be aware of the DetNet forwarding sub-layer.

In general, if a DetNet flow passes through one or more DetNet-unaware network nodes between two DetNet nodes providing the DetNet forwarding sub-layer for that flow, there is a potential for disruption or failure of the DetNet QoS. A network administrator needs to ensure that the DetNet-unaware network nodes are configured to minimize the chances of packet loss and delay, and provision enough extra buffer space in the DetNet transit node following the DetNet-unaware network nodes to absorb the induced latency variations.

#### 4.3. DetNet flows

##### 4.3.1. DetNet flow types

A DetNet flow can have different formats while its packets are forwarded between the peer end systems depending on the type of the end systems. Corresponding to the end system types, the following possible types / formats of a DetNet flow are distinguished in this document. The different flow types have different requirements to DetNet nodes.

- o App-flow: the payload (data) carried over a DetNet flow between DetNet unaware end systems. An app-flow does not contain any DetNet related attributes and does not imply any specific requirement on DetNet nodes.
- o DetNet-f-flow: specific format of a DetNet flow. It only requires the resource allocation features provided by the DetNet forwarding sub-layer.
- o DetNet-s-flow: specific format of a DetNet flow. It only requires the service protection feature ensured by the DetNet service sub-layer.
- o DetNet-sf-flow: specific format of a DetNet flow. It requires both DetNet service sub-layer and DetNet forwarding sub-layer functions during forwarding.

##### 4.3.2. Source transmission behavior

For the purposes of resource allocation, DetNet flows can be synchronous or asynchronous. In synchronous DetNet flows, at least the DetNet nodes (and possibly the end systems) are closely time synchronized, typically to better than 1 microsecond. By transmitting packets from different DetNet flows or classes of DetNet

flows at different times, using repeating schedules synchronized among the DetNet nodes, resources such as buffers and link bandwidth can be shared over the time domain among different DetNet flows. There is a tradeoff among techniques for synchronous DetNet flows between the burden of fine-grained scheduling and the benefit of reducing the required resources, especially buffer space.

In contrast, asynchronous DetNet flows are not coordinated with a fine-grained schedule, so relay and end systems must assume worst-case interference among DetNet flows contending for buffer resources. Asynchronous DetNet flows are characterized by:

- o A maximum packet size;
- o An observation interval; and
- o A maximum number of transmissions during that observation interval.

These parameters, together with knowledge of the protocol stack used (and thus the size of the various headers added to a packet), provide the bandwidth that is needed for the DetNet flow.

The source is required not to exceed these limits in order to obtain DetNet service. If the source transmits less data than this limit allows, the unused resource such as link bandwidth can be made available by the DetNet system to non-DetNet packets as long as all guarantees are fulfilled. However, making those resources available to DetNet packets in other DetNet flows would serve no purpose. Those other DetNet flows have their own dedicated resources, on the assumption that all DetNet flows can use all of their resources over a long period of time.

There is no expectation in DetNet for App-flows to be responsive to congestion control [RFC2914] or explicit congestion notification [RFC3168]. The assumption is that a DetNet flow, to be useful, must be delivered in its entirety. That is, while any useful application is written to expect a certain number of lost packets, the real-time applications of interest to DetNet demand that the loss of data due to the network is a rare event.

Although DetNet strives to minimize the changes required of an application to allow it to shift from a special-purpose digital network to an Internet Protocol network, one fundamental shift in the behavior of network applications is impossible to avoid: the reservation of resources before the application starts. In the first place, a network cannot deliver finite latency and practically zero packet loss to an arbitrarily high offered load. Secondly, achieving

practically zero packet loss for DetNet flows means that DetNet nodes have to dedicate buffer resources to specific DetNet flows or to classes of DetNet flows. The requirements of each reservation have to be translated into the parameters that control each DetNet system's queuing, shaping, and scheduling functions and delivered to the DetNet nodes and end systems.

All nodes in a DetNet domain are expected to support the data behavior required to deliver a particular DetNet service. If a node itself is not DetNet service aware, the DetNet nodes that are adjacent to such non-DetNet aware nodes must ensure that the non-DetNet aware node is provisioned to appropriately support the DetNet service. For example, an IEEE 802.1 TSN node may be used to interconnect DetNet aware nodes, and these DetNet nodes can map DetNet flows to 802.1 TSN flows. Another example, an MPLS-TE or TP domain may be used to interconnect DetNet aware nodes, and these DetNet nodes can map DetNet flows to TE LSPs which can provide the QoS requirements of the DetNet service.

#### 4.3.3. Incomplete Networks

The presence in the network of intermediate nodes or subnets that are not fully capable of offering DetNet services complicates the ability of the intermediate nodes and/or controller to allocate resources, as extra buffering must be allocated at points downstream from the non-DetNet intermediate node for a DetNet flow. This extra buffering may increase latency and/or jitter.

#### 4.4. Traffic Engineering for DetNet

Traffic Engineering Architecture and Signaling (TEAS) [TEAS] defines traffic-engineering architectures for generic applicability across packet and non-packet networks. From a TEAS perspective, Traffic Engineering (TE) refers to techniques that enable operators to control how specific traffic flows are treated within their networks.

Because of its very nature of establishing explicit optimized paths, Deterministic Networking can be seen as a new, specialized branch of Traffic Engineering, and inherits its architecture with a separation into planes.

The Deterministic Networking architecture is thus composed of three planes, a (User) Application Plane, a Controller Plane, and a Network Plane, which echoes that of Figure 1 of Software-Defined Networking (SDN): Layers and Architecture Terminology [RFC7426], and the Controllers identified in [RFC8453] and [RFC7149].



#### 4.4.1. The Application Plane

Per [RFC7426], the Application Plane includes both applications and services. In particular, the Application Plane incorporates the User Agent, a specialized application that interacts with the end user / operator and performs requests for Deterministic Networking 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. An abstraction of the flow called a Traffic Specification (TSpec) provides the representation. This abstraction is used to place a reservation over the (Northbound) Service Interface and within the Application plane. It is associated with an abstraction of location, such as IP addresses and DNS names, to identify the end systems and possibly specify DetNet nodes.

#### 4.4.2. The Controller Plane

The Controller Plane corresponds to the aggregation of the Control and Management Planes in [RFC7426], though Common Control and Measurement Plane (CCAMP) [CCAMP] makes an additional distinction between management and measurement. When the logical separation of the Control, Measurement and other Management entities is not relevant, the term Controller Plane is used for simplicity to represent them all, and the term Controller Plane Function (CPF) refers to any device operating in that plane, whether is it a Path Computation Element (PCE) [RFC4655], or a Network Management entity (NME), or a distributed control plane. 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 as illustrated in Figure 7.

One or more CPF(s) collaborate to implement the requests from the FME as Per-Flow Per-Hop Behaviors installed in the DetNet nodes for each individual flow. The CPFs place each flow along a deterministic sequence of DetNet nodes so as to respect per-flow constraints such as security and latency, and optimize the overall result for metrics such as an abstract aggregated cost. The deterministic sequence can typically be more complex than a direct sequence and include redundant paths, with one or more packet replication and elimination points. Scaling to larger networks is discussed in Section 4.9.

#### 4.4.3. The Network 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 Forwarding Plane (data plane), Application, and Operational Plane (e.g., OAM) aspects.

The network Plane comprises the Network Interface Cards (NIC) in the end systems, which are typically IP hosts, and DetNet nodes, which are typically IP routers and MPLS switches. Network-to-Network Interfaces such as used for Traffic Engineering path reservation in [RFC5921], as well as User-to-Network Interfaces (UNI) such as provided by the Local Management Interface (LMI) between network and end systems, are both part of the Network Plane, both in the control plane and the data plane.

A Southbound (Network) Interface enables the entities in the Controller Plane to communicate with devices in the Network Plane as illustrated in Figure 7. This interface leverages and extends TEAS to describe the physical topology and resources in the Network Plane.

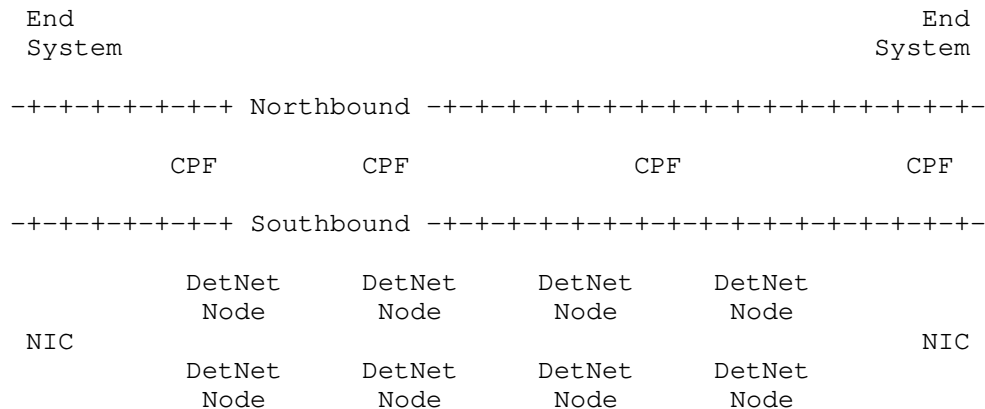


Figure 7: Northbound and Southbound interfaces

The DetNet nodes (and possibly the end systems NIC) expose their capabilities and physical resources to the controller (the CPF), and update the CPFs with their dynamic perception of the topology, across the Southbound Interface. In return, the CPFs set the per-flow paths up, providing a Flow Characterization that is more tightly coupled to the DetNet node Operation than a TSpec.

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, and forward packets within constraints

associated to each flow, or, when unable to do so, perform a last resort operation such as drop or declassify.

This document focuses on the Southbound interface and the operation of the Network Plane.

#### 4.5. Queuing, Shaping, Scheduling, and Preemption

DetNet achieves bounded delivery latency by reserving bandwidth and buffer resources at each DetNet node along the path of the DetNet flow. The reservation itself is not sufficient, however. Implementors and users of a number of proprietary and standard real-time networks have found that standards for specific data plane techniques are required to enable these assurances to be made in a multi-vendor network. The fundamental reason is that latency variation in one DetNet system results in the need for extra buffer space in the next-hop DetNet system(s), which in turn, increases the worst-case per-hop latency.

Standard queuing and transmission selection algorithms allow traffic engineering Section 4.4 to compute the latency contribution of each DetNet node to the end-to-end latency, to compute the amount of buffer space required in each DetNet node for each incremental DetNet flow, and most importantly, to translate from a flow specification to a set of values for the managed objects that control each relay or end system. For example, the IEEE 802.1 WG has specified (and is specifying) a set of queuing, shaping, and scheduling algorithms that enable each DetNet node, and/or a central controller, to compute these values. These algorithms include:

- o A credit-based shaper [IEEE802.1Qav] (superseded by [IEEE802.1Q-2018]).
- o Time-gated queues governed by a rotating time schedule based on synchronized time [IEEE802.1Qbv] (superseded by [IEEE802.1Q-2018]).
- o Synchronized double (or triple) buffers driven by synchronized time ticks. [IEEE802.1Qch] (superseded by [IEEE802.1Q-2018]).
- o Pre-emption of an Ethernet packet in transmission by a packet with a more stringent latency requirement, followed by the resumption of the preempted packet [IEEE802.1Qbu] (superseded by [IEEE802.1Q-2018]), [IEEE802.3br] (superseded by [IEEE802.3-2018]).

While these techniques are currently embedded in Ethernet [IEEE802.3-2018] and bridging standards, we can note that they are

all, except perhaps for packet preemption, equally applicable to other media than Ethernet, and to routers as well as bridges. Other media may have its own methods, see, e.g., [I-D.ietf-6tisch-architecture], [RFC7554]. Further techniques are defined by the IETF, e.g., [RFC8289] and [RFC8033]. DetNet may include such definitions in the future, or may define how these techniques can be used by DetNet nodes.

#### 4.6. Service instance

A Service instance represents all the functions required on a DetNet node to allow the end-to-end service between the UNIs.

The DetNet network general reference model is shown in Figure 8 for a DetNet service scenario (i.e., between two DetNet-UNIs). In this figure, end systems ("A" and "B") are connected directly to the edge nodes of an IP/MPLS network ("PE1" and "PE2"). End systems participating in DetNet communication may require connectivity before setting up an App-flow that requires the DetNet service. Such a connectivity related service instance and the one dedicated for DetNet service share the same access. Packets belonging to a DetNet flow are selected by a filter configured on the access ("F1" and "F2"). As a result, data flow specific access ("access-A + F1" and "access-B + F2") are terminated in the flow specific service instance ("SI-1" and "SI-2"). A tunnel is used to provide connectivity between the service instances.

The tunnel is exclusively used for the packets of the DetNet flow between "SI-1" and "SI-2". The service instances are configured to implement DetNet functions and a flow specific DetNet forwarding. The service instance and the tunnel may or may not be shared by multiple DetNet flows. Sharing the service instance by multiple DetNet flows requires properly populated forwarding tables of the service instance.

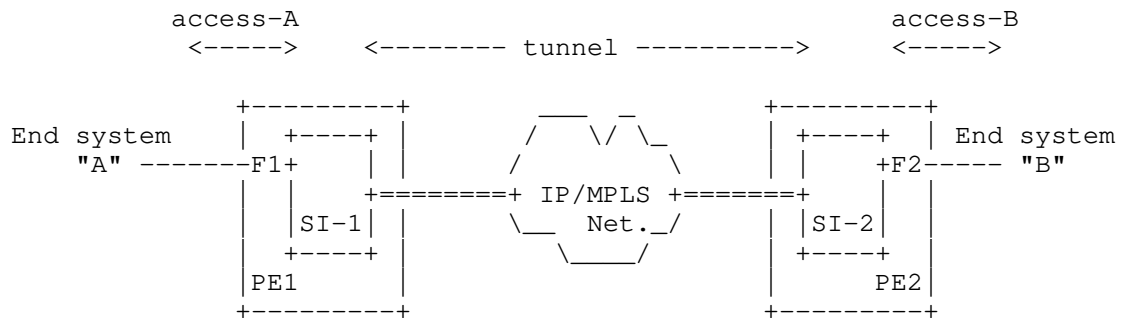


Figure 8: DetNet network general reference model

The tunnel between the service instances may have some special characteristics. For example, in case of a DetNet L3 service, there are differences in the usage of the PW for DetNet traffic compared to the network model described in [RFC6658]. In the DetNet scenario, the PW is likely to be used exclusively by the DetNet flow, whereas [RFC6658] states: "The packet PW appears as a single point-to-point link to the client layer. Network-layer adjacency formation and maintenance between the client equipment will follow the normal practice needed to support the required relationship in the client layer ... This packet PseudoWire is used to transport all of the required Layer-2 and Layer-3 protocols between LSR1 and LSR2". Further details are network technology specific and can be found in [I-D.ietf-detnet-dp-sol-mpls] and [I-D.ietf-detnet-dp-sol-ip].

#### 4.7. Flow identification at technology borders

This section discusses what needs to be done at technology borders including Ethernet as one of the technologies. Flow identification for MPLS and IP data planes are described in [I-D.ietf-detnet-dp-sol-mpls] and [I-D.ietf-detnet-dp-sol-ip], respectively.

##### 4.7.1. Exporting flow identification

A DetNet node may need to map specific flows to lower layer flows (or Streams) in order to provide specific queuing and shaping services for specific flows. For example:

- o A non-IP, strictly L2 source end system X may be sending multiple flows to the same L2 destination end system Y. Those flows may include DetNet flows with different QoS requirements, and may include non-DetNet flows.

- o A router may be sending any number of flows to another router. Again, those flows may include DetNet flows with different QoS requirements, and may include non-DetNet flows.
- o Two routers may be separated by bridges. For these bridges to perform any required per-flow queuing and shaping, they must be able to identify the individual flows.
- o A Label Edge Router (LER) may have a Label Switched Path (LSP) set up for handling traffic destined for a particular IP address carrying only non-DetNet flows. If a DetNet flow to that same address is requested, a separate LSP may be needed, in order that all of the Label Switch Routers (LSRs) along the path to the destination give that flow special queuing and shaping.

The need for a lower-layer node to be aware of individual higher-layer flows is not unique to DetNet. But, given the endless complexity of layering and relayering over tunnels that is available to network designers, DetNet needs to provide a model for flow identification that is better than packet inspection. That is not to say that packet inspection to Layer-4 or Layer-5 addresses will not be used, or the capability standardized; but, there are alternatives.

A DetNet relay node can connect DetNet flows on different paths using different flow identification methods. For example:

- o A single unicast DetNet flow passing from router A through a bridged network to router B may be assigned a TSN Stream identifier that is unique within that bridged network. The bridges can then identify the flow without accessing higher-layer headers. Of course, the receiving router must recognize and accept that TSN Stream.
- o A DetNet flow passing from LSR A to LSR B may be assigned a different label than that used for other flows to the same IP destination.

In any of the above cases, it is possible that an existing DetNet flow can be an aggregate carrying multiple other DetNet flows. (Not to be confused with DetNet compound vs. member flows.) Of course, this requires that the aggregate DetNet flow be provisioned properly to carry the aggregated flows.

Thus, rather than packet inspection, there is the option to export higher-layer information to the lower layer. The requirement to support one or the other method for flow identification (or both) is a complexity that is part of DetNet control models.

#### 4.7.2. Flow attribute mapping between layers

Forwarding of packets of DetNet flows over multiple technology domains may require that lower layers are aware of specific flows of higher layers. Such an "exporting of flow identification" is needed each time when the forwarding paradigm is changed on the forwarding path (e.g., two LSRs are interconnected by a L2 bridged domain, etc.). The three representative forwarding methods considered for deterministic networking are:

- o IP routing
- o MPLS label switching
- o Ethernet bridging

A packet with corresponding Flow-IDs is illustrated in Figure 9, which also indicates where each Flow-ID can be added or removed.

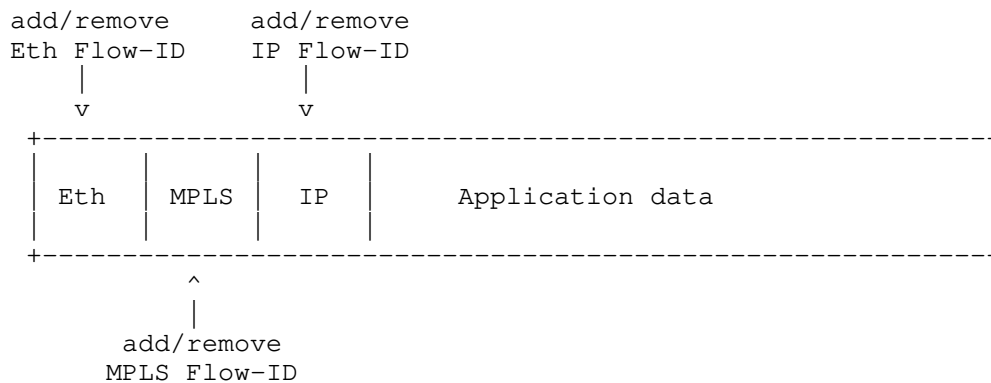


Figure 9: Packet with multiple Flow-IDs

The additional (domain specific) Flow-ID can be

- o created by a domain specific function or
- o derived from the Flow-ID added to the App-flow.

The Flow-ID must be unique inside a given domain. Note that the Flow-ID added to the App-flow is still present in the packet, but some nodes may lack the function to recognize it; that's why the additional Flow-ID is added.

## 4.7.3. Flow-ID mapping examples

IP nodes and MPLS nodes are assumed to be configured to push such an additional (domain specific) Flow-ID when sending traffic to an Ethernet switch (as shown in the examples below).

Figure 10 shows a scenario where an IP end system ("IP-A") is connected via two Ethernet switches ("ETH-n") to an IP router ("IP-1").

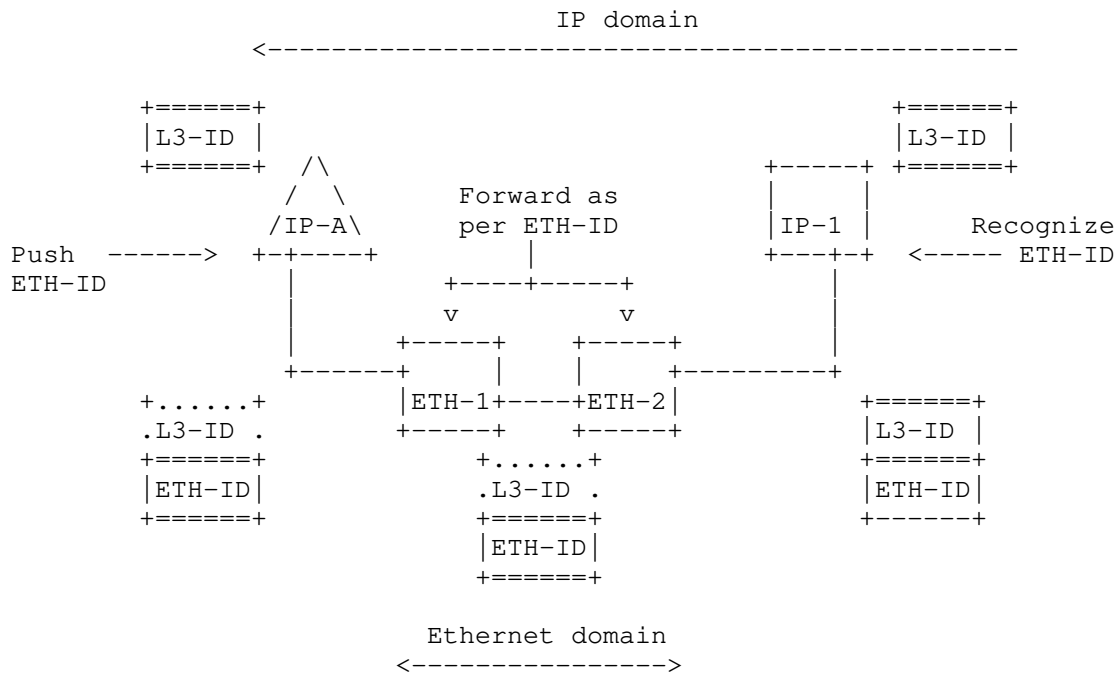


Figure 10: IP nodes interconnected by an Ethernet domain

End system "IP-A" uses the original App-flow specific ID ("L3-ID"), but as it is connected to an Ethernet domain it has to push an Ethernet-domain specific flow-ID ("ETH-ID") before sending the packet to "ETH-1" node. Ethernet switch "ETH-1" can recognize the data flow based on the "ETH-ID" and it does forwarding toward "ETH-2". "ETH-2" switches the packet toward the IP router. "IP-1" must be configured to receive the Ethernet Flow-ID specific multicast flow, but (as it is an L3 node) it decodes the data flow ID based on the "L3-ID" fields of the received packet.

Figure 11 shows a scenario where MPLS domain nodes ("PE-n" and "P-m") are connected via two Ethernet switches ("ETH-n").



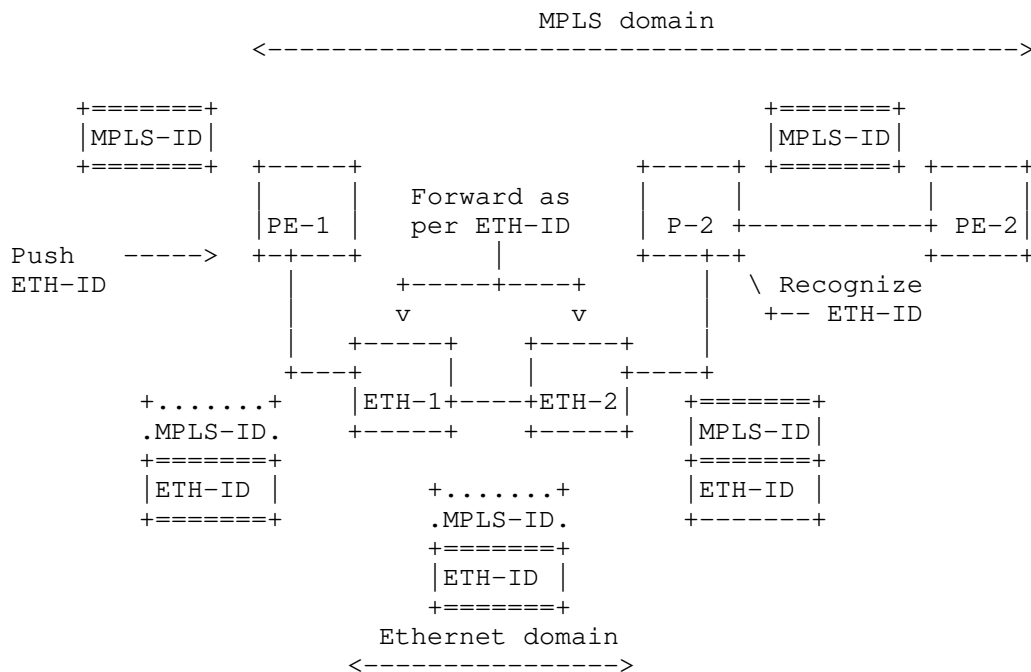


Figure 11: MPLS nodes interconnected by an Ethernet domain

"PE-1" uses the MPLS specific ID ("MPLS-ID"), but as it is connected to an Ethernet domain it has to push an Ethernet-domain specific flow-ID ("ETH-ID") before sending the packet to "ETH-1". Ethernet switch "ETH-1" can recognize the data flow based on the "ETH-ID" and it does forwarding toward "ETH-2". "ETH-2" switches the packet toward the MPLS node ("P-2"). "P-2" must be configured to receive the Ethernet Flow-ID specific multicast flow, but (as it is an MPLS node) it decodes the data flow ID based on the "MPLS-ID" fields of the received packet.

One can appreciate from the above example that, when the means used for DetNet flow identification is altered or exported, the means for encoding the sequence number information must similarly be altered or exported.

#### 4.8. Advertising resources, capabilities and adjacencies

Provisioning of DetNet requires knowledge about:

- o Details of the DetNet system's capabilities that are required in order to accurately allocate that DetNet system's resources, as well as other DetNet systems' resources. This includes, for

example, which specific queuing and shaping algorithms are implemented (Section 4.5), the number of buffers dedicated for DetNet allocation, and the worst-case forwarding delay and misordering.

- o The actual state of a DetNet node's DetNet resources.
- o The identity of the DetNet system's neighbors, and the characteristics of the link(s) between the DetNet systems, including the latency of the links (in nanoseconds).

#### 4.9. Scaling to larger networks

Reservations for individual DetNet flows require considerable state information in each DetNet node, especially when adequate fault mitigation (Section 3.3.2) is required. The DetNet data plane, in order to support larger numbers of DetNet flows, must support the aggregation of DetNet flows. Such aggregated flows can be viewed by the DetNet nodes' data plane largely as individual DetNet flows. Without such aggregation, the per-relay system may limit the scale of DetNet networks. Example techniques that may be used include MPLS hierarchy and IP DiffServ Code Points (DSCPs).

#### 4.10. Compatibility with Layer-2

Standards providing similar capabilities for bridged networks (only) have been and are being generated in the IEEE 802 LAN/MAN Standards Committee. The present architecture describes an abstract model that can be applicable both at Layer-2 and Layer-3, and over links not defined by IEEE 802.

DetNet enabled end systems and DetNet nodes can be interconnected by sub-networks, i.e., Layer-2 technologies. These sub-networks will provide DetNet compatible service for support of DetNet traffic. Examples of sub-network technologies include MPLS TE, 802.1 TSN, and a point-to-point OTN link. Of course, multi-layer DetNet systems may be possible too, where one DetNet appears as a sub-network, and provides service to, a higher layer DetNet system.

### 5. Security Considerations

Security considerations for DetNet are described in detail in [I-D.ietf-detnet-security]. This section considers exclusively security considerations which are specific to the DetNet architecture.

Security aspects which are unique to DetNet are those whose aim is to provide the specific quality of service aspects of DetNet, which are

primarily to deliver data flows with extremely low packet loss rates and bounded end-to-end delivery latency. A DetNet may be implemented using MPLS and/or IP (including both v4 and v6) technologies, and thus inherits the security properties of those technologies at both the data plane and the control plane.

Security considerations for DetNet are constrained (compared to, for example, the open Internet) because DetNet is defined to operate only within a single administrative domain (see Section 1). The primary considerations are to secure the request and control of DetNet resources, maintain confidentiality of data traversing the DetNet, and provide the availability of the DetNet quality of service.

To secure the request and control of DetNet resources, authentication and authorization can be used for each device connected to a DetNet domain, most importantly to network controller devices. Control of a DetNet network may be centralized or distributed (within a single administrative domain). In the case of centralized control, precedent for security considerations as defined for Abstraction and Control of Traffic Engineered Networks (ACTN) can be found in [RFC8453], Section 9. In the case of distributed control protocols, DetNet security is expected to be provided by the security properties of the protocols in use. In any case, the result is that manipulation of administratively configurable parameters is limited to authorized entities.

To maintain confidentiality of data traversing the DetNet, application flows can be protected through whatever means is provided by the underlying technology. For example, encryption may be used, such as that provided by IPSec [RFC4301] for IP flows and by MACSec [IEEE802.1AE-2018] for Ethernet (Layer-2) flows.

DetNet flows are identified on a per-flow basis, which may provide attackers with additional information about the data flows (when compared to networks that do not include per-flow identification). This is an inherent property of DetNet which has security implications that should be considered when determining if DetNet is a suitable technology for any given use case.

To provide uninterrupted availability of the DetNet quality of service, provisions can be made against DOS attacks and delay attacks. To protect against DOS attacks, excess traffic due to malicious or malfunctioning devices can be prevented or mitigated, for example through the use of traffic admission control applied at the input of a DetNet domain, as described in Section 3.2.1, and through the fault mitigation methods described in Section 3.3.2. To prevent DetNet packets from being delayed by an entity external to a DetNet domain, DetNet technology definition can allow for the

mitigation of Man-In-The-Middle attacks, for example through use of authentication and authorization of devices within the DetNet domain.

Because DetNet mechanisms or applications that rely on DetNet can make heavy use of methods that require precise time synchronization, the accuracy, availability, and integrity of time synchronization is of critical importance. Extensive discussion of this topic can be found in [RFC7384].

DetNet use cases are known to have widely divergent security requirements. The intent of this section is to provide a baseline for security considerations which are common to all DetNet designs and implementations, without burdening individual designs with specifics of security infrastructure which may not be germane to the given use case. Designers and implementers of DetNet systems are expected to take use case specific considerations into account in their DetNet designs and implementations.

## 6. Privacy Considerations

DetNet provides a Quality of Service (QoS), and as such, is not expected to directly raise any new privacy considerations, the generic considerations for such mechanisms apply. In particular, such markings allow for an attacker to correlate flows or to select particular types of flow for more detailed inspection.

However, the requirement for every (or almost every) node along the path of a DetNet flow to identify DetNet flows may present an additional attack surface for privacy, should the DetNet paradigm be found useful in broader environments.

## 7. IANA Considerations

This document does not require an action from IANA.

## 8. Acknowledgements

The authors wish to thank Lou Berger, David Black, Stewart Bryant, Rodney Cummings, Ethan Grossman, Craig Gunther, Marcel Kiessling, Rudy Klecka, Jouni Korhonen, Erik Nordmark, Shitanshu Shah, Wilfried Steiner, George Swallow, Michael Johas Teener, Pat Thaler, Thomas Watteyne, Patrick Wetterwald, Karl Weber, Anca Zamfir, for their various contributions to this work.

## 9. Informative References

## [BUFFERBLOAT]

Gettys, J. and K. Nichols, "Bufferbloat: Dark Buffers in the Internet", January 2012.

## [CCAMP]

IETF, "Common Control and Measurement Plane Working Group",  
<<https://datatracker.ietf.org/doc/charter-ietf-ccamp/>>.

## [I-D.ietf-6tisch-architecture]

Thubert, P., "An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4", draft-ietf-6tisch-architecture-20 (work in progress), March 2019.

## [I-D.ietf-detnet-dp-sol-ip]

Korhonen, J. and B. Varga, "DetNet IP Data Plane Encapsulation", draft-ietf-detnet-dp-sol-ip-01 (work in progress), October 2018.

## [I-D.ietf-detnet-dp-sol-mpls]

Korhonen, J. and B. Varga, "DetNet MPLS Data Plane Encapsulation", draft-ietf-detnet-dp-sol-mpls-01 (work in progress), October 2018.

## [I-D.ietf-detnet-problem-statement]

Finn, N. and P. Thubert, "Deterministic Networking Problem Statement", draft-ietf-detnet-problem-statement-09 (work in progress), December 2018.

## [I-D.ietf-detnet-security]

Mizrahi, T., Grossman, E., Hacker, A., Das, S., Dowdell, J., Austad, H., Stanton, K., and N. Finn, "Deterministic Networking (DetNet) Security Considerations", draft-ietf-detnet-security-04 (work in progress), March 2019.

## [I-D.ietf-detnet-use-cases]

Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-20 (work in progress), December 2018.

## [IEC62439-3-2016]

International Electrotechnical Commission (IEC) TC 65/SC 65C - Industrial networks, "IEC 62439-3:2016 Industrial communication networks - High availability automation networks - Part 3: Parallel Redundancy Protocol (PRP) and High-availability Seamless Redundancy (HSR)", 2016, <<https://webstore.iec.ch/publication/24447>>.

- [IEEE802.1AE-2018]  
IEEE Standards Association, "IEEE Std 802.1AE-2018 MAC Security (MACsec)", 2018,  
<<https://ieeexplore.ieee.org/document/8585421>>.
- [IEEE802.1BA]  
IEEE Standards Association, "IEEE Std 802.1BA-2011 Audio Video Bridging (AVB) Systems", 2011,  
<<https://ieeexplore.ieee.org/document/6032690>>.
- [IEEE802.1CB]  
IEEE Standards Association, "IEEE Std 802.1CB-2017 Frame Replication and Elimination for Reliability", 2017,  
<<https://ieeexplore.ieee.org/document/8091139>>.
- [IEEE802.1Q-2018]  
IEEE Standards Association, "IEEE Std 802.1Q-2018 Bridges and Bridged Networks", 2018,  
<<https://ieeexplore.ieee.org/document/8403927>>.
- [IEEE802.1Qav]  
IEEE Standards Association, "IEEE Std 802.1Qav-2009 Bridges and Bridged Networks - Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams", 2009, <<https://ieeexplore.ieee.org/document/5375704>>.
- [IEEE802.1Qbu]  
IEEE Standards Association, "IEEE Std 802.1Qbu-2016 Bridges and Bridged Networks - Amendment 26: Frame Preemption", 2016,  
<<https://ieeexplore.ieee.org/document/7553415>>.
- [IEEE802.1Qbv]  
IEEE Standards Association, "IEEE Std 802.1Qbv-2015 Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic", 2015,  
<<https://ieeexplore.ieee.org/document/7572858>>.
- [IEEE802.1Qch]  
IEEE Standards Association, "IEEE Std 802.1Qch-2017 Bridges and Bridged Networks - Amendment 29: Cyclic Queuing and Forwarding", 2017,  
<<https://ieeexplore.ieee.org/document/7961303>>.
- [IEEE802.1TSNTG]  
IEEE Standards Association, "IEEE 802.1 Time-Sensitive Networking Task Group", <<http://www.ieee802.org/1/tsn>>.

- [IEEE802.3-2018]  
IEEE Standards Association, "IEEE Std 802.3-2018 Standard for Ethernet", 2018,  
<<https://ieeexplore.ieee.org/document/8457469>>.
- [IEEE802.3br]  
IEEE Standards Association, "IEEE Std 802.3br-2016 Standard for Ethernet Amendment 5: Specification and Management Parameters for Interspersing Express Traffic", 2016, <<http://ieeexplore.ieee.org/document/7900321/>>.
- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<https://www.rfc-editor.org/info/rfc2205>>.
- [RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, DOI 10.17487/RFC2475, December 1998, <<https://www.rfc-editor.org/info/rfc2475>>.
- [RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, DOI 10.17487/RFC2914, September 2000, <<https://www.rfc-editor.org/info/rfc2914>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <<https://www.rfc-editor.org/info/rfc4655>>.

- [RFC5921] Bocci, M., Ed., Bryant, S., Ed., Frost, D., Ed., Levrau, L., and L. Berger, "A Framework for MPLS in Transport Networks", RFC 5921, DOI 10.17487/RFC5921, July 2010, <<https://www.rfc-editor.org/info/rfc5921>>.
- [RFC6372] Sprecher, N., Ed. and A. Farrel, Ed., "MPLS Transport Profile (MPLS-TP) Survivability Framework", RFC 6372, DOI 10.17487/RFC6372, September 2011, <<https://www.rfc-editor.org/info/rfc6372>>.
- [RFC6658] Bryant, S., Ed., Martini, L., Swallow, G., and A. Malis, "Packet Pseudowire Encapsulation over an MPLS PSN", RFC 6658, DOI 10.17487/RFC6658, July 2012, <<https://www.rfc-editor.org/info/rfc6658>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.
- [RFC7384] Mizrahi, T., "Security Requirements of Time Protocols in Packet Switched Networks", RFC 7384, DOI 10.17487/RFC7384, October 2014, <<https://www.rfc-editor.org/info/rfc7384>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<https://www.rfc-editor.org/info/rfc7426>>.
- [RFC7554] Watteyne, T., Ed., Palattella, M., and L. Grieco, "Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement", RFC 7554, DOI 10.17487/RFC7554, May 2015, <<https://www.rfc-editor.org/info/rfc7554>>.
- [RFC7813] Farkas, J., Ed., Bragg, N., Unbehagen, P., Parsons, G., Ashwood-Smith, P., and C. Bowers, "IS-IS Path Control and Reservation", RFC 7813, DOI 10.17487/RFC7813, June 2016, <<https://www.rfc-editor.org/info/rfc7813>>.
- [RFC8033] Pan, R., Natarajan, P., Baker, F., and G. White, "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem", RFC 8033, DOI 10.17487/RFC8033, February 2017, <<https://www.rfc-editor.org/info/rfc8033>>.



- [RFC8227] Cheng, W., Wang, L., Li, H., van Helvoort, H., and J. Dong, "MPLS-TP Shared-Ring Protection (MSRP) Mechanism for Ring Topology", RFC 8227, DOI 10.17487/RFC8227, August 2017, <<https://www.rfc-editor.org/info/rfc8227>>.
- [RFC8289] Nichols, K., Jacobson, V., McGregor, A., Ed., and J. Iyengar, Ed., "Controlled Delay Active Queue Management", RFC 8289, DOI 10.17487/RFC8289, January 2018, <<https://www.rfc-editor.org/info/rfc8289>>.
- [RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8453] Ceccarelli, D., Ed. and Y. Lee, Ed., "Framework for Abstraction and Control of TE Networks (ACTN)", RFC 8453, DOI 10.17487/RFC8453, August 2018, <<https://www.rfc-editor.org/info/rfc8453>>.
- [TEAS] IETF, "Traffic Engineering Architecture and Signaling Working Group", <<https://datatracker.ietf.org/doc/charter-ietf-teas/>>.

## Authors' Addresses

Norman Finn  
Huawei  
3101 Rio Way  
Spring Valley, California 91977  
US

Phone: +1 925 980 6430  
Email: [norman.finn@mail01.huawei.com](mailto:norman.finn@mail01.huawei.com)

Pascal Thubert  
Cisco Systems  
Village d'Entreprises Green Side  
400, Avenue de Roumanille  
Batiment T3  
Biot - Sophia Antipolis 06410  
FRANCE

Phone: +33 4 97 23 26 34  
Email: [pthubert@cisco.com](mailto:pthubert@cisco.com)

Balazs Varga  
Ericsson  
Magyar tudosok korutja 11  
Budapest 1117  
Hungary

Email: balazs.a.varga@ericsson.com

Janos Farkas  
Ericsson  
Magyar tudosok korutja 11  
Budapest 1117  
Hungary

Email: janos.farkas@ericsson.com

DetNet  
Internet-Draft  
Intended status: Standards Track  
Expires: September 11, 2019

J. Korhonen, Ed.  
B. Varga, Ed.  
Ericsson  
March 10, 2019

DetNet IP Data Plane Encapsulation  
draft-ietf-detnet-dp-sol-ip-02

Abstract

This document specifies the Deterministic Networking data plane when operating in an IP packet switched network.

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 September 11, 2019.

Copyright Notice

Copyright (c) 2019 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 . . . . .	3
2. Terminology . . . . .	3
2.1. Terms Used In This Document . . . . .	3
2.2. Abbreviations . . . . .	4
2.3. Requirements Language . . . . .	4
3. DetNet IP Data Plane Overview . . . . .	5
4. DetNet IP Data Plane Considerations . . . . .	7
4.1. End-System Specific Considerations . . . . .	8
4.2. DetNet Domain-Specific Considerations . . . . .	9
4.2.1. DetNet Routers . . . . .	10
4.3. Networks With Multiple Technology Segments . . . . .	11
4.4. OAM . . . . .	12
4.5. Class of Service . . . . .	12
4.6. Quality of Service . . . . .	13
4.7. Cross-DetNet Flow Resource Aggregation . . . . .	14
4.8. Time Synchronization . . . . .	14
5. Management and Control Considerations . . . . .	15
5.1. Flow Identification and Aggregation . . . . .	15
5.2. Explicit Routes . . . . .	16
5.3. Contention Loss and Jitter Reduction . . . . .	16
5.4. Bidirectional Traffic . . . . .	17
5.5. DetNet Controller (Control and Management) Plane Requirements . . . . .	17
6. DetNet IP Data Plane Procedures . . . . .	19
6.1. DetNet IP Flow Identification Procedures . . . . .	19
6.1.1. IP Header Information . . . . .	19
6.1.2. Other Protocol Header Information . . . . .	21
6.1.3. Flow Identification Management and Control Information . . . . .	22
6.2. Forwarding Procedures . . . . .	23
6.3. DetNet IP Traffic Treatment Procedures . . . . .	23
6.4. Aggregation Considerations . . . . .	23
7. IP over DetNet MPLS . . . . .	24
7.1. IP Over DetNet MPLS Data Plane Scenarios . . . . .	24
7.2. DetNet IP over DetNet MPLS Encapsulation . . . . .	27
7.3. DetNet IP over DetNet MPLS Flow Identification Procedures . . . . .	29
7.4. DetNet IP over DetNet MPLS Traffic Treatment Procedures . . . . .	29
8. Mapping DetNet IP Flows to IEEE 802.1 TSN . . . . .	29
8.1. TSN Stream ID Mapping . . . . .	31
8.2. TSN Usage of FRER . . . . .	33
8.3. Procedures . . . . .	34
8.4. Management and Control Implications . . . . .	34
9. Security Considerations . . . . .	36
10. IANA Considerations . . . . .	36
11. Contributors . . . . .	36

12. Acknowledgements . . . . .	38
13. References . . . . .	38
13.1. Normative references . . . . .	38
13.2. Informative references . . . . .	40
Appendix A. Example of DetNet Data Plane Operation . . . . .	43
Appendix B. Example of Pinned Paths Using IPv6 . . . . .	43
Authors' Addresses . . . . .	43

## 1. Introduction

Deterministic Networking (DetNet) is a service that can be offered by a network to DetNet flows. DetNet provides these flows extremely low packet loss rates and assured maximum end-to-end delivery latency. General background and concepts of DetNet can be found in the DetNet Architecture [I-D.ietf-detnet-architecture].

This document specifies the DetNet data plane operation for IP hosts and routers that provide DetNet service to IP encapsulated data. No DetNet specific encapsulation is defined to support IP flows, rather existing IP and higher layer protocol header information is used to support flow identification and DetNet service delivery.

The DetNet Architecture decomposes the DetNet related data plane functions into two sub-layers: a service sub-layer and a forwarding sub-layer. The service sub-layer is used to provide DetNet service protection and reordering. The forwarding sub-layer is used to provide congestion protection (low loss, assured latency, and limited reordering). As no DetNet specific headers are added to support DetNet IP flows, only the forwarding sub-layer functions are supported using the DetNet IP defined by this document. Service protection can be provided on a per sub-net basis using technologies such as MPLS [I-D.ietf-detnet-dp-sol-mpls] and IEEE802.1 TSN.

This document provides an overview of the DetNet IP data plane in Section 3, considerations that apply to providing DetNet services via the DetNet IP data plane in Section 4 and Section 5. Section 6 provides the procedures for hosts and routers that support IP-based DetNet services. Finally, Section 8 provides rules for mapping IP-based DetNet flows to IEEE 802.1 TSN streams.

## 2. Terminology

### 2.1. Terms Used In This Document

This document uses the terminology and concepts established in the DetNet architecture [I-D.ietf-detnet-architecture], and the reader is assumed to be familiar with that document and its terminology.

## 2.2. Abbreviations

The following abbreviations used in this document:

CE	Customer Edge equipment.
CoS	Class of Service.
DetNet	Deterministic Networking.
DF	DetNet Flow.
L2	Layer-2.
L3	Layer-3.
LSP	Label-switched path.
MPLS	Multiprotocol Label Switching.
OAM	Operations, Administration, and Maintenance.
PE	Provider Edge.
PREOF	Packet Replication, Ordering and Elimination Function.
PSN	Packet Switched Network.
PW	Pseudowire.
QoS	Quality of Service.
TE	Traffic Engineering.
TSN	Time-Sensitive Networking, TSN is a Task Group of the IEEE 802.1 Working Group.

## 2.3. 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 IP Data Plane Overview

This document describes how IP is used by DetNet nodes, i.e., hosts and routers, to identify DetNet flows and provide a DetNet service. From a data plane perspective, an end-to-end IP model is followed. As mentioned above, existing IP and higher layer protocol header information is used to support flow identification and DetNet service delivery.

DetNet uses "6-tuple" based flow identification, where "6-tuple" refers to information carried in IP and higher layer protocol headers. General background on the use of IP headers, and "5-tuples", to identify flows and support Quality of Service (QoS) can be found in [RFC3670]. [RFC7657] also provides useful background on the delivery differentiated services (DiffServ) and "6-tuple" based flow identification.

DetNet flow aggregation may be enabled via the use of wildcards, masks, prefixes and ranges. IP tunnels may also be used to support flow aggregation. In these cases, it is expected that DetNet aware intermediate nodes will provide DetNet service assurance on the aggregate through resource allocation and congestion control mechanisms.

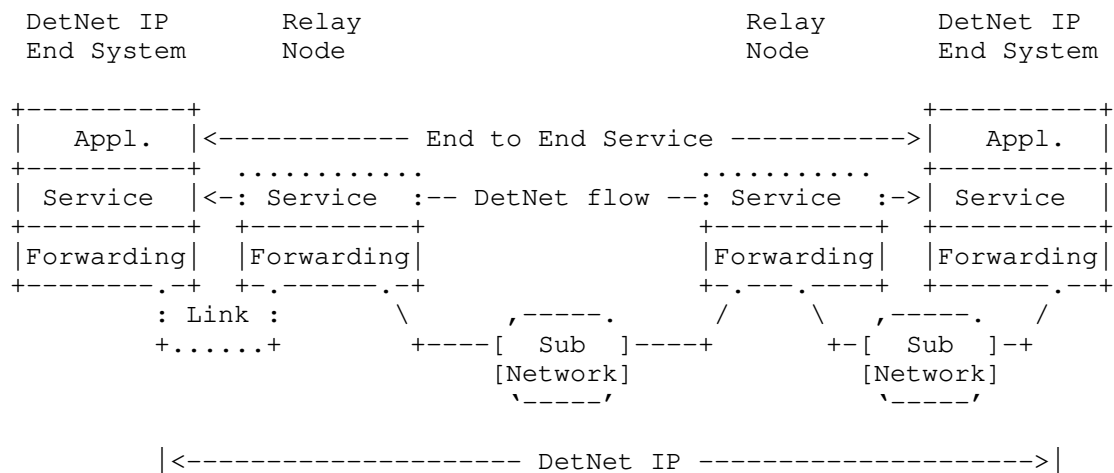


Figure 1: A Simple DetNet (DN) Enabled IP Network

Figure 1 illustrates a DetNet enabled IP network. The DetNet enabled end systems originate IP encapsulated traffic that is identified as DetNet flows, relay nodes understand the forwarding requirements of the DetNet flow and ensure that node, interface and sub-network resources are allocated to ensure DetNet service requirements. The

dotted line around the Service component of the Relay Nodes indicates that the transit routers are DetNet service aware but do not perform any DetNet service sub-layer function, e.g., PREOF. IEEE 802.1 TSN is an example sub-network type which can provide support for DetNet flows and service. The mapping of DetNet IP flows to TSN streams and TSN protection mechanisms is covered in Section 8.

Note: The sub-network can represent a TSN, MPLS or IP network segment.

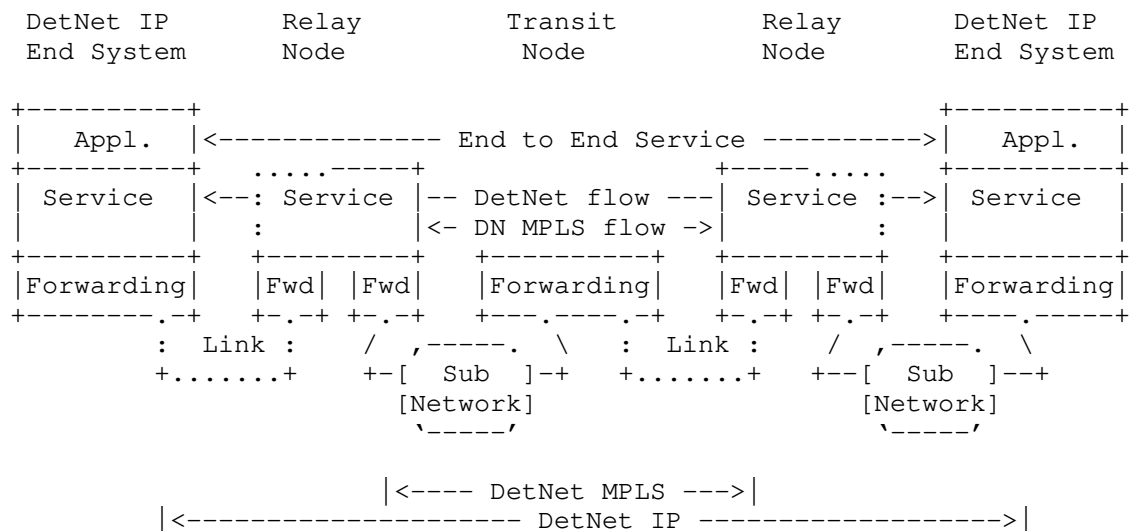


Figure 2: DetNet IP Over DetNet MPLS Network

Figure 2 illustrates a variant of Figure 1, with an MPLS based DetNet network as a sub-network between the relay nodes. It shows a more complex DetNet enabled IP network where an IP flow is mapped to one or more PWs and MPLS (TE) LSPs. The end systems still originate IP encapsulated traffic that is identified as DetNet flows. The relay nodes follow procedures defined in Section 7 to map each DetNet flow to MPLS LSPs. While not shown, relay nodes can provide service sub-layer functions such as PREOF using DetNet over MPLS, and this is indicated by the solid line for the MPLS facing portion of the Service component. Note that the Transit node is MPLS (TE) LSP aware and performs switching based on MPLS labels, and need not have any specific knowledge of the DetNet service or the corresponding DetNet flow identification. See Section 7 for details on the mapping of IP flows to MPLS, and [I-D.ietf-detnet-dp-sol-mpls] for general support of DetNet services using MPLS.



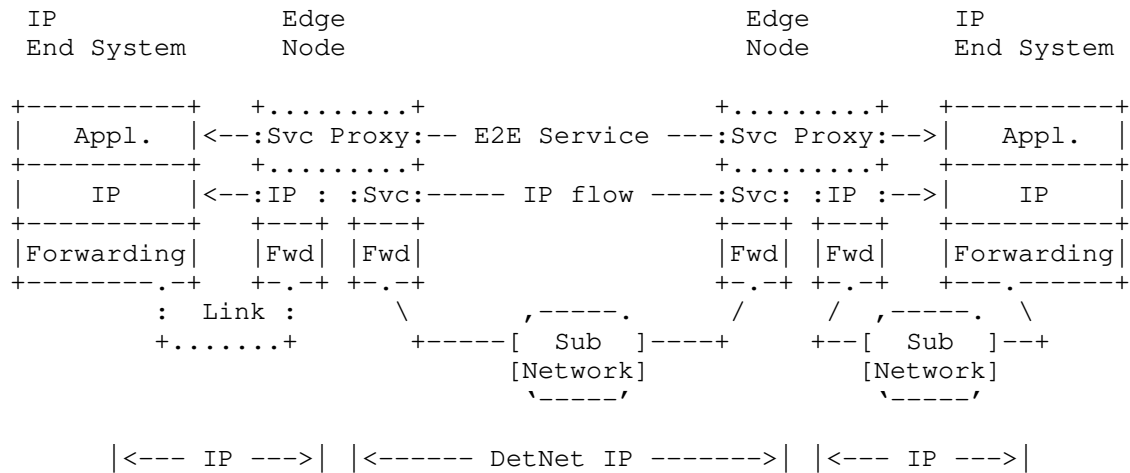


Figure 3: Non-DetNet aware IP end systems with DetNet IP Domain

Figure 3 illustrates another variant of Figure 1 where the end systems are not DetNet aware. In this case, edge nodes sit at the boundary of the DetNet domain and provide DetNet service proxies for the end applications by initiating and terminating DetNet service for the application's IP flows. The existing header information or an approach such as described in Section 4.7 can be used to support DetNet flow identification.

Non-DetNet and DetNet IP packets are identical on the wire. From data plane perspective, the only difference is that there is flow-associated DetNet information on each DetNet node that defines the flow related characteristics and required forwarding behavior. As shown above, edge nodes provide a Service Proxy function that "associates" one or more IP flows with the appropriate DetNet flow-specific information and ensures that the receives the proper traffic treatment within the domain.

Note: The operation of IEEE802.1 TSN end systems over DetNet enabled IP networks is not described in this document. While TSN flows could be encapsulated in IP packets by an IP End System or DetNet Edge Node in order to produce DetNet IP flows, the details of such are out of scope of this document.

#### 4. DetNet IP Data Plane Considerations

This section provides informative considerations related to providing DetNet service to flows which are identified based on their header information. At a high level, the following are provided on a per flow basis:

#### Congestion protection and latency control:

Usage of allocated resources (queuing, policing, shaping) to ensure that the congestion-related loss and latency/jitter requirements of a DetNet flow are met.

#### Explicit routes:

Use of a specific path for a flow. This limits misordering and can improve delivery of deterministic latency.

#### Service protection:

Which in the case of this document translates to changing the explicit path after a failure is detected in order to restore delivery of the required DetNet service characteristics. Path changes, even in the case of failure recovery, can lead to the out of order delivery of data.

Note: DetNet PREOF is not provided by the mechanisms defined in this document.

#### Load sharing:

Generally, distributing packets of the same DetNet flow over multiple paths is not recommended. Such load sharing, e.g., via ECMP or UCMP, impacts ordering and end-to-end jitter.

#### Troubleshooting:

For example, to support identification of misbehaving flows.

#### Recognize flow(s) for analytics:

For example, increase counters.

#### Correlate events with flows:

For example, unexpected loss.

### 4.1. End-System Specific Considerations

Data-flows requiring DetNet service are generated and terminated on end systems. This document deals only with IP end systems. The protocols used by an IP end system are specific to an application and end systems peer with end systems using the same application encapsulation format. This said, DetNet's use of 6-tuple IP flow identification means that DetNet must be aware of not only the format

of the IP header, but also of the next protocol carried within an IP packet.

When IP end systems are DetNet aware, no application-level or service-level proxy functions are needed inside the DetNet domain. For DetNet unaware IP end systems service-level proxy functions are needed inside the DetNet domain.

End systems need to ensure that DetNet service requirements are met when processing packets associated with a DetNet flow. When forwarding packets, this means that packets are appropriately shaped on transmission and received appropriate traffic treatment on the connected sub-network, see Section 4.6 and Section 4.2.1 for more details. When receiving packets, this means that there are appropriate local node resources, e.g., buffers, to receive and process a DetNet flow packets.

#### 4.2. DetNet Domain-Specific Considerations

As a general rule, DetNet IP domains need to be able to forward any DetNet flow identified by the IP 6-tuple. Doing otherwise would limit end system encapsulation format. From a practical standpoint this means that all nodes along the end-to-end path of a DetNet flows need to agree on what fields are used for flow identification, and the transport protocols (e.g., TCP/UDP/IPsec) which can be used to identify 6-tuple protocol ports.

From a connection type perspective two scenarios are identified:

1. DN attached: end system is directly connected to an edge node or end system is behind a sub-network. (See ES1 and ES2 in figure below)
2. DN integrated: end system is part of the DetNet domain. (See ES3 in figure below)

L3 (IP) end systems may use any of these connection types. DetNet domain allows communication between any end-systems using the same encapsulation format, independent of their connection type and DetNet capability. DN attached end systems have no knowledge about the DetNet domain and its encapsulation format. See Figure 4 for L3 end system connection scenarios.

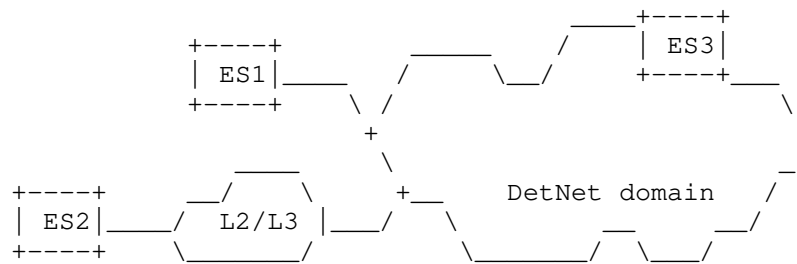


Figure 4: Connection types of L3 end systems

#### 4.2.1. DetNet Routers

Within a DetNet domain, the DetNet enabled IP Routers interconnect links and sub-networks to support end-to-end delivery of DetNet flows. From a DetNet architecture perspective, these routers are DetNet relays, as they must be DetNet service aware. Such routers identify DetNet flows based on the IP 6-tuple, and ensure that the DetNet service required traffic treatment is provided both on the node and on any attached sub-network.

This solution provides DetNet functions end to end, but does so on a per link and sub-network basis. Congestion protection and latency control and the resource allocation (queuing, policing, shaping) are supported using the underlying link / sub net specific mechanisms. However, service protections (packet replication and packet elimination functions) are not provided at the DetNet layer end to end. But such service protection can be provided on a per underlying L2 link and sub-network basis.

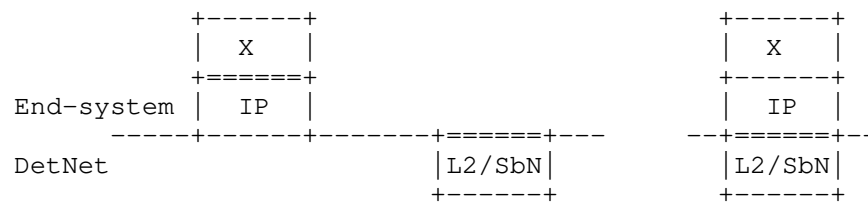


Figure 5: Encapsulation of DetNet Routing in simplified IP service L3 end-systems

The DetNet Service Flow is mapped to the link / sub-network specific resources using an underlying system specific means. This implies each DetNet aware node on path looks into the forwarded DetNet

Service Flow packet and utilize e.g., a 5- (or 6-) tuple to find out the required mapping within a node.

As noted earlier, the Service Protection is done within each link / sub-network independently using the domain specific mechanisms (due the lack of a unified end to end sequencing information that would be available for intermediate nodes). Therefore, service protection (if any) cannot be provided end-to-end, only within sub-networks. This is shown for a three sub-network scenario in Figure 6, where each sub-network can provide service protection between its borders.

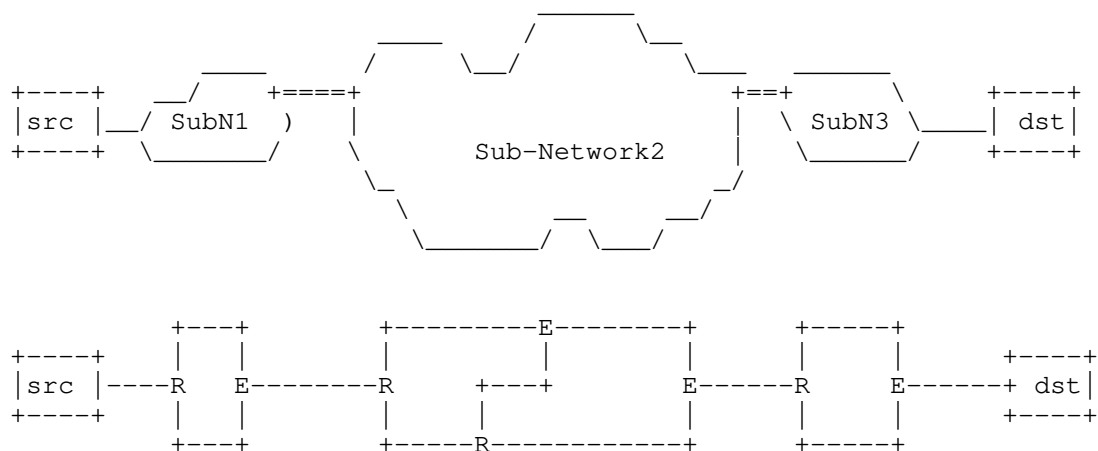


Figure 6: Replication and elimination in sub-networks for DetNet IP networks

If end to end service protection is desired that can be implemented, for example, by the DetNet end systems using Layer-4 (L4) transport protocols or application protocols. However, these are out of scope of this document.

#### 4.3. Networks With Multiple Technology Segments

There are network scenarios, where the DetNet domain contains multiple technology segments (IEEE 802.1 TSN, MPLS) and all those segments are under the same administrative control (see Figure 7). Furthermore, DetNet nodes may be interconnected via TSN segments.

DetNet routers ensure that detnet service requirements are met per hop by allocating local resources, both receive and transmit, and by mapping the service requirements of each flow to appropriate sub-

network mechanisms. Such mapping is sub-network technology specific. The mapping of DetNet IP Flows to MPLS is covered Section 7. The mapping of IP DetNet Flows to IEEE 802.1 TSN is covered in Section 8.

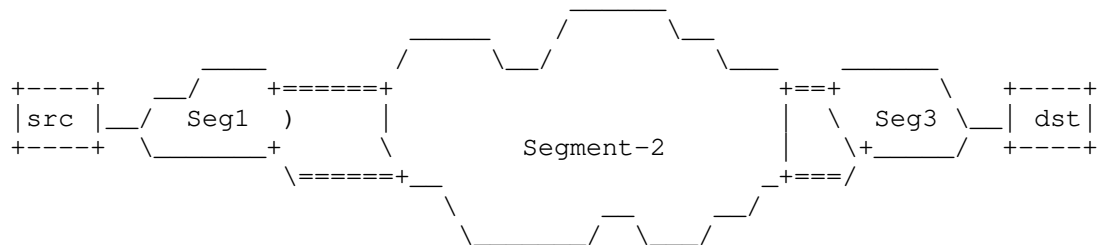


Figure 7: DetNet domains and multiple technology segments

#### 4.4. OAM

[Editor's note: This section is TBD. OAM may be dropped from this document and left for future study.]

#### 4.5. Class of Service

Class and quality of service, i.e., CoS and QoS, are terms that are often used interchangeably and confused. In the context of DetNet, CoS is used to refer to mechanisms that provide traffic forwarding treatment based on aggregate group basis and QoS is used to refer to mechanisms that provide traffic forwarding treatment based on a specific DetNet flow basis. Examples of existing network level CoS mechanisms include DiffServ which is enabled by IP header differentiated services code point (DSCP) field [RFC2474] and MPLS label traffic class field [RFC5462], and at Layer-2, by IEEE 802.1p priority code point (PCP).

CoS for DetNet flows carried in IPv6 is provided using the standard differentiated services code point (DSCP) field [RFC2474] and related mechanisms. The 2-bit explicit congestion notification (ECN) [RFC3168] field MAY also be used.

One additional consideration for DetNet nodes which support CoS services is that they MUST ensure that the CoS service classes do not impact the congestion protection and latency control mechanisms used to provide DetNet QoS. This requirement is similar to requirement for MPLS LSRs to that CoS LSPs do not impact the resources allocated to TE LSPs via [RFC3473].

#### 4.6. Quality of Service

Quality of Service (QoS) mechanisms for flow-specific traffic treatment typically includes a guarantee/agreement for the service, and allocation of resources to support the service. Example QoS mechanisms include discrete resource allocation, admission control, flow identification and isolation, and sometimes path control, traffic protection, shaping, policing and remarking. Example protocols that support QoS control include Resource ReSerVation Protocol (RSVP) [RFC2205] (RSVP) and RSVP-TE [RFC3209] and [RFC3473]. The existing MPLS mechanisms defined to support CoS [RFC3270] can also be used to reserve resources for specific traffic classes.

In addition to explicit routes, and packet replication and elimination, DetNet provides zero congestion loss and bounded latency and jitter. As described in [I-D.ietf-detnet-architecture], there are different mechanisms that maybe used separately or in combination to deliver a zero congestion loss service. These mechanisms are provided by the either the MPLS or IP layers, and may be combined with the mechanisms defined by the underlying network layer such as 802.1TSN.

A baseline set of QoS capabilities for DetNet flows carried in PWs and MPLS can provided by MPLS with Traffic Engineering (MPLS-TE) [RFC3209] and [RFC3473]. TE LSPs can also support explicit routes (path pinning). Current service definitions for packet TE LSPs can be found in "Specification of the Controlled Load Quality of Service", [RFC2211], "Specification of Guaranteed Quality of Service", [RFC2212], and "Ethernet Traffic Parameters", [RFC6003]. Additional service definitions are expected in future documents to support the full range of DetNet services. In all cases, the existing label-based marking mechanisms defined for TE-LSPs and even E-LSPs are use to support the identification of flows requiring DetNet QoS.

QoS for DetNet service flows carried in IP MUST be provided locally by the DetNet-aware hosts and routers supporting DetNet flows. Such support will leverage the underlying network layer such as 802.1TSN. The traffic control mechanisms used to deliver QoS for IP encapsulated DetNet flows are expected to be defined in a future document. From an encapsulation perspective, the combination of the "6 tuple" i.e., the typical 5 tuple enhanced with the DSCP code, uniquely identifies a DetNet service flow.

Packets that are marked with a DetNet Class of Service value, but that have not been the subject of a completed reservation, can disrupt the QoS offered to properly reserved DetNet flows by using

resources allocated to the reserved flows. Therefore, the network nodes of a DetNet network must:

- o Defend the DetNet QoS by discarding or remarking (to a non-DetNet CoS) packets received that are not the subject of a completed reservation.
- o Not use a DetNet reserved resource, e.g. a queue or shaper reserved for DetNet flows, for any packet that does not carry a DetNet Class of Service marker.

#### 4.7. Cross-DetNet Flow Resource Aggregation

The ability to aggregate individual flows, and their associated resource control, into a larger aggregate is an important technique for improving scaling of control in the data, management and control planes. This document identifies the traffic identification related aspects of aggregation of DetNet flows. The resource control and management aspects of aggregation (including the queuing/shaping/policing implications) will be covered in other documents. The data plane implications of aggregation are independent for PW/MPLS and IP encapsulated DetNet flows.

DetNet flows forwarded via IP have more limited aggregation options, due to the available traffic flow identification fields of the IP solution. One available approach is to manage the resources associated with a DSCP identified traffic class and to map (remark) individually controlled DetNet flows onto that traffic class. This approach also requires that nodes support aggregation ensure that traffic from aggregated LSPs are placed (shaped/policed/enqueued) in a fashion that ensures the required DetNet service is preserved.

In both the MPLS and IP cases, additional details of the traffic control capabilities needed at a DetNet-aware node may be covered in the new service descriptions mentioned above or in separate future documents. Management and control plane mechanisms will also need to ensure that the service required on the aggregate flow (H-LSP or DSCP) are provided, which may include the discarding or remarking mentioned in the previous sections.

#### 4.8. Time Synchronization

While time synchronization can be important both from the perspective of operating the DetNet network itself and from the perspective of DetNet-based applications, time synchronization is outside the scope of this document. This said, a DetNet node can also support time synchronization or distribution mechanisms.



For example, [RFC8169] describes a method of recording the packet queuing time in an MPLS LSR on a packet by per packet basis and forwarding this information to the egress edge system. This allows compensation for any variable packet queuing delay to be applied at the packet receiver. Other mechanisms for IP networks are defined based on IEEE Standard 1588 [IEEE1588], such as ITU-T [G.8275.1] and [G.8275.2].

A more detailed discussion of time synchronization is outside the scope of this document.

## 5. Management and Control Considerations

While management plane and control planes are traditionally considered separately, from the Data Plane perspective there is no practical difference based on the origin of flow provisioning information, and the DetNet architecture [I-D.ietf-detnet-architecture] refers to these collectively as the 'Controller Plane'. This document therefore does not distinguish between information provided by distributed control plane protocols, e.g., IGP routing protocols, or by centralized network management mechanisms, e.g., RestConf [RFC8040], YANG [RFC7950], and the Path Computation Element Communication Protocol (PCEP) [RFC8283] [I-D.ietf-teas-pce-native-ip] or any combination thereof. Specific considerations and requirements for the DetNet Controller Plane are discussed in Section 5.5.

### 5.1. Flow Identification and Aggregation

Section 3 introduces the use of the IP "6-tuple" for flow identification, and Section 4.6 goes on to discuss how identified flows use specific QoS mechanisms for flow-specific traffic treatment, including path control and resource allocation. Section 6.1 contains detailed DetNet IP flow identification procedures. Flow identification will play an important role for the DetNet controller plane.

Section 4.7 and Section 6.4 discuss the use of flow aggregation in DetNet. Flow aggregation can be accomplished using any of the 6-tuple fields defined in Section 6.1, using a DSCP identified traffic class or other field. It will be the responsibility of the DetNet controller plane to be able to properly provision the use of these aggregation mechanisms. These requirements are included in Section 5.5.

## 5.2. Explicit Routes

Explicit routes are used to ensure that packets are routed through the resources that have been reserved for them, and hence provide the DetNet application with the required service. A requirement for the DetNet Controller Plane will be the ability to assign a particular identified DetNet IP flow to a path through the DetNet domain that has been assigned the required nodal resources to provide the appropriate traffic treatment for the flow, and also to include particular links as a part of the path that are able to support the DetNet flow, for example by using IEEE 802.1 TSN links (as discussed in Section 8). Further considerations and requirements for the DetNet Controller Plane are discussed in Section 5.5.

Whether configuring, calculating and instantiating these routes is a single-stage or multi-stage process, or in a centralized or distributed manner, is out of scope of this document.

There are several of approaches that could be used to provide explicit routes and resource allocation in the DetNet layer. For example:

- o The path could be explicitly set up by a controller which calculates the path and explicitly configures each node along that path with the appropriate forwarding and resource allocation information.
- o The path could be used a distributed control plane such as RSVP [RFC2205] or RSVP-TE [RFC3473] extended to support DetNet IP flows.
- o The path could be implemented using IPv6-based segment routing when extended to support resource allocation.

See Section 5.5 for further discussion of these alternatives. In addition, [RFC2386] contains useful background information on QoS-based routing, and [RFC5575] discusses a specific mechanism used by BGP for traffic flow specification and policy-based routing.

## 5.3. Contention Loss and Jitter Reduction

As discussed in Section 1, this document does not specify the mechanisms needed to eliminate contention loss or reduce jitter for DetNet flows at the DetNet forwarding sub-layer. The ability to manage node and link resources to be able to provide these functions will be a necessary part of the DetNet controller plane. It will also be necessary to be able to control the required queuing mechanisms used to provide these functions along a flow's path

through the network. See Section 6.3 and Section 5.5 for further discussion of these requirements.

#### 5.4. Bidirectional Traffic

Some DetNet applications generate bidirectional traffic. Although this document discusses the DetNet IP data plane, MPLS definitions [RFC5654] are useful to illustrate terms such as associated bidirectional flows and co-routed bidirectional flows. MPLS defines a point-to-point associated bidirectional LSP as consisting of two unidirectional point-to-point LSPs, one from A to B and the other from B to A, which are regarded as providing a single logical bidirectional forwarding path. This is analogous to standard IP routing. MPLS defines a point-to-point co-routed bidirectional LSP as an associated bidirectional LSP which satisfies the additional constraint that its two unidirectional component LSPs follow the same path (in terms of both nodes and links) in both directions. An important property of co-routed bidirectional LSPs is that their unidirectional component LSPs share fate. In both types of bidirectional LSPs, resource reservations may differ in each direction. The concepts of associated bidirectional flows and co-routed bidirectional flows can also be applied to DetNet IP flows.

While the DetNet IP data plane must support bidirectional DetNet flows, there are no special bidirectional features with respect to the data plane other than the need for the two directions of a co-routed bidirectional flow to take the same path. That is to say that bidirectional DetNet flows are solely represented at the management and control plane levels, without specific support or knowledge within the DetNet data plane. Fate sharing and associated vs. co-routed bidirectional flows can be managed at the control level.

Control and management mechanisms will need to support bidirectional flows, but the specification of such mechanisms are out of scope of this document. An example control plane solution for MPLS can be found in [RFC7551].

This is further discussed in Section 5.5.

#### 5.5. DetNet Controller (Control and Management) Plane Requirements

While the definition of controller plane for DetNet is out of the scope of this document, there are particular considerations and requirements for such that result from the unique characteristics of the DetNet architecture [I-D.ietf-detnet-architecture] and data plane as defined herein.

The primary requirements of the DetNet controller plane are that it must be able to:

- o Instantiate DetNet flows in a DetNet domain (which may include some or all of explicit path determination, link bandwidth reservations, restricting flows to IEEE 802.1 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.
- o The ability to support DetNet flow aggregation
- o 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)
- o Scale to handle the number of DetNet flows expected in a domain (which may require per-flow signaling or provisioning)
- o Provision flow identification information at each of the nodes along the path, and it may differ depending on the location in the network and the DetNet functionality.

These requirements, as stated earlier, could be satisfied using distributed control protocol signaling, centralized network management provisioning mechanisms, or hybrid combinations of the two, and could also make use of IPv6-based segment routing.

In the abstract, the results of either distributed signaling or centralized provisioning are equivalent from a DetNet data plane perspective – flows are instantiated, explicit routes are determined, resources are reserved, and packets are forwarded through the domain using the IP data plane.

However, from a practical and implementation standpoint, they are not equivalent at all. Some approaches are more scalable than others in terms of signaling load on the network. Some can take advantage of global tracking of resources in the DetNet domain for better overall network resource optimization. Some are more resilient than others if link, node, or management equipment failures occur. While a detailed analysis of the control plane alternatives is out of the scope of this document, the requirements from this document can be used as the basis of a later analysis of the alternatives.

## 6. DetNet IP Data Plane Procedures

This section provides DetNet IP data plane procedures. These procedures have been divided into the following areas: flow identification, forwarding and traffic treatment. Flow identification includes those procedures related to matching IP and higher layer protocol header information to DetNet flow (state) information and service requirements. Flow identification is also sometimes called Traffic classification, for example see [RFC5777]. Forwarding includes those procedures related to next hop selection and delivery. Traffic treatment includes those procedures related to providing an identified flow with the required DetNet service.

DetNet IP data plane procedures also have implications on the control and management of DetNet flows and these are also covered in this section. Specifically this section identifies a number of information elements that will require support via the management and control interfaces supported by a DetNet node. The specific mechanism used for such support is out of the scope of this document. A summary of the management and control related information requirements is included. Conformance language is not used in the summary as it applies to future mechanisms such as those that may be provided in YANG models [YANG-REF-TBD].

### 6.1. DetNet IP Flow Identification Procedures

IP and higher layer protocol header information is used to identify DetNet flows. All DetNet implementations that support this document MUST identify individual DetNet flows based on the set of information identified in this section. Note, that additional flow identification requirements, e.g., to support other higher layer protocols, may be defined in future.

The configuration and control information used to identify an individual DetNet flow MUST be ordered by an implementation. Implementations MUST support a fixed order when identifying flows, and MUST identify a DetNet flow by the first set of matching flow information.

Implementations of this document MUST support DetNet flow identification when the implementation is acting as a DetNet end systems, a relay node or as an edge node.

#### 6.1.1. IP Header Information

Implementations of this document MUST support DetNet flow identification based on IP header information. The IPv4 header is defined in [RFC0791] and the IPv6 is defined in [RFC8200].

#### 6.1.1.1. Source Address Field

Implementations of this document MUST support DetNet flow identification based on the Source Address field of an IP packet. Implementations SHOULD support longest prefix matching for this field, see [RFC1812] and [RFC7608]. Note that a prefix length of zero (0) effectively means that the field is ignored.

#### 6.1.1.2. Destination Address Field

Implementations of this document MUST support DetNet flow identification based on the Destination Address field of an IP packet. Implementations SHOULD support longest prefix matching for this field, see [RFC1812] and [RFC7608]. Note that a prefix length of zero (0) effectively means that the field is ignored.

Note: any IP address value is allowed, including IP multicast destination address.

#### 6.1.1.3. IPv4 Protocol and IPv6 Next Header Fields

Implementations of this document MUST support DetNet flow identification based on the IPv4 Protocol field when processing IPv4 packets, and the IPv6 Next Header Field when processing IPv6 packets. An implementation MUST support flow identification based on the next protocol values defined in Section 6.1.2. Other, non-zero values, MUST be used for flow identification. Implementations SHOULD allow for these fields to be ignored for a specific DetNet flow.

#### 6.1.1.4. IPv4 Type of Service and IPv6 Traffic Class Fields

These fields are used to support Differentiated Services [RFC2474] and Explicit Congestion Notification [RFC3168]. Implementations of this document MUST support DetNet flow identification based on the IPv4 Type of Service field when processing IPv4 packets, and the IPv6 Traffic Class Field when processing IPv6 packets. Implementations MUST support bitmask based matching, where one (1) values in the bitmask indicate which subset of the bits in the field are to be used in determining a match. Note that a zero (0) value as a bitmask effectively means that these fields are ignored.

#### 6.1.1.5. IPv6 Flow Label Field

Implementations of this document SHOULD support identification of DetNet flows based on the IPv6 Flow Label field. Implementations that support matching based on this field MUST allow for this field to be ignored for a specific DetNet flow. When this field is used to identify a specific DetNet flow, implementations MAY exclude the

IPv6 Next Header field and next header information as part of DetNet flow identification.

#### 6.1.2. Other Protocol Header Information

Implementations of this document MUST support DetNet flow identification based on header information identified in this section. Support for TCP, UDP and IPsec flows are defined. Future documents are expected to define support for other protocols.

##### 6.1.2.1. TCP and UDP

DetNet flow identification for TCP [RFC0793] and UDP [RFC0768] is done based on the Source and Destination Port fields carried in each protocol's header. These fields share a common format and common DetNet flow identification procedures.

###### 6.1.2.1.1. Source Port Field

Implementations of this document MUST support DetNet flow identification based on the Source Port field of a TCP or UDP packet. Implementations MUST support flow identification based on a particular value carried in the field, i.e., an exact. Implementations SHOULD support range-based port matching. Implementation MUST also allow for the field to be ignored for a specific DetNet flow.

###### 6.1.2.1.2. Destination Port Field

Implementations of this document MUST support DetNet flow identification based on the Destination Port field of a TCP or UDP packet. Implementations MUST support flow identification based on a particular value carried in the field, i.e., an exact. Implementations SHOULD support range-based port matching. Implementation MUST also allow for the field to be ignored for a specific DetNet flow.

##### 6.1.2.2. IPsec AH and ESP

IPsec Authentication Header (AH) [RFC4302] and Encapsulating Security Payload (ESP) [RFC4303] share a common format for the Security Parameters Index (SPI) field. Implementations MUST support flow identification based on a particular value carried in the field, i.e., an exact. Implementation SHOULD also allow for the field to be ignored for a specific DetNet flow.

### 6.1.3. Flow Identification Management and Control Information

The following summarizes the set of information that is needed to identify an individual DetNet flow:

- o IPv4 and IPv6 source address field.
- o IPv4 and IPv6 source address prefix length, where a zero (0) value effectively means that the address field is ignored.
- o IPv4 and IPv6 destination address field.
- o IPv4 and IPv6 destination address prefix length, where a zero (0) effectively means that the address field is ignored.
- o IPv4 protocol field. A limited set of values is allowed, and the ability to ignore this field, e.g., via configuration of the value zero (0), is desirable.
- o IPv6 next header field. A limited set of values is allowed, and the ability to ignore this field, e.g., via configuration of the value zero (0), is desirable.
- o IPv4 Type of Service and IPv6 Traffic Class Fields.
- o IPv4 Type of Service and IPv6 Traffic Class Field Bitmask, where a zero (0) effectively means that these fields are ignored.
- o IPv6 flow label field. This field can be optionally used for matching. When used, can be exclusive of matching against the next header field.
- o TCP and UDP Source Port. Exact and wildcard matching is required. Port ranges can optionally be used.
- o TCP and UDP Destination Port. Exact and wildcard matching is required. Port ranges can optionally be used.

This information MUST be provisioned per DetNet flow via configuration, e.g., via the controller plane described in Section 5.

Information identifying a DetNet flow is ordered and implementations use the first match. This can, for example, be used to provide a DetNet service for a specific UDP flow, with unique Source and Destination Port field values, while providing a different service for all other flows with that same UDP Destination Port value.



## 6.2. Forwarding Procedures

General requirements for IP nodes are defined in [RFC1122], [RFC1812] and [RFC6434], and are not modified by this document. The typical next-hop selection process is impacted by DetNet. Specifically, implementations of this document SHALL use management and control information to select the one or more outgoing interfaces and next hops to be used for a packet belonging to a DetNet flow.

The use of multiple paths or links, e.g., ECMP, to support a single DetNet flow is NOT RECOMMENDED. ECMP MAY be used for non-DetNet flows within a DetNet domain.

The above implies that management and control functions will be defined to support this requirement, e.g., see [YANG-REF-TBD].

## 6.3. DetNet IP Traffic Treatment Procedures

Implementations of this document MUST ensure that a DetNet flow receives the traffic treatment that is provisioned for it via configuration or the controller plane, e.g., via [YANG-REF-TBD]. General information on DetNet service can be found in [I-D.ietf-detnet-flow-information-model]. Typical mechanisms used to provide different treatment to different flows includes the allocation of system resources (such as queues and buffers) and provisioning or related parameters (such as shaping, and policing). Support can also be provided via an underlying network technology such as MPLS Section 7 and IEEE802.1 TSN Section 8. Other than in the TSN case, the specific mechanisms used by a DetNet node to ensure DetNet service delivery requirements are met for supported DetNet flows is outside the scope of this document.

## 6.4. Aggregation Considerations

The use of prefixes, wildcards, bitmasks, and port ranges allows a DetNet node to aggregate DetNet flows. This aggregation can take place within a single node, when that node maintains state about both the aggregated and component flows. It can also take place between nodes, where one node maintains state about only flow aggregates while the other node maintains state on all or a portion of the component flows. In either case, the management or control function that provisions the aggregate flows must ensure that adequate resources are allocated and configured to provide combined service requirements of the component flows. As DetNet is concerned about latency and jitter, more than just bandwidth needs to be considered.

## 7. IP over DetNet MPLS

This section defines how IP encapsulated flows are carried over a DetNet MPLS data plane as defined in [I-D.ietf-detnet-dp-sol-mpls]. As Non-DetNet and DetNet IP packets are identical on the wire, this section is applicable to any node that supports IP over DetNet MPLS, and this section refers to both cases as DetNet IP over DetNet MPLS.

### 7.1. IP Over DetNet MPLS Data Plane Scenarios

This section provides example uses of IP over DetNet MPLS for illustrative purposes.

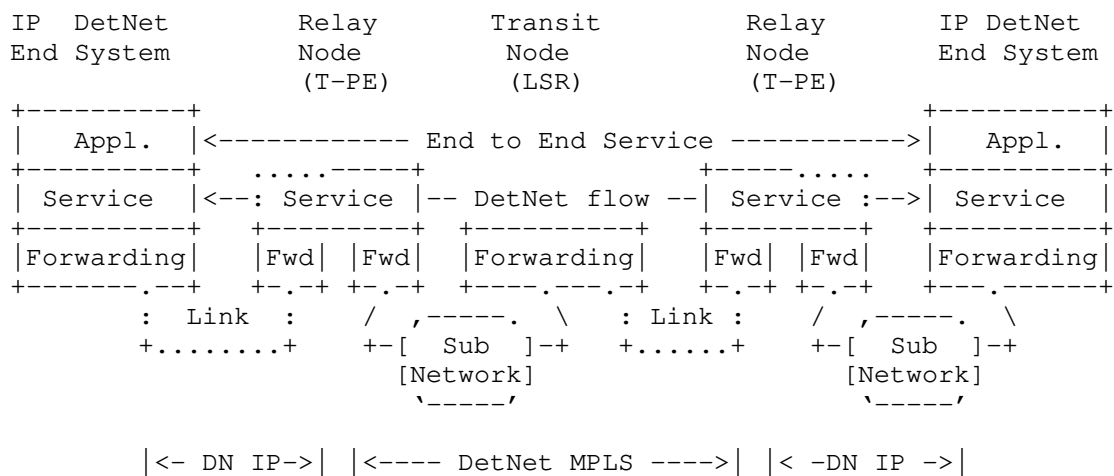


Figure 8: DetNet IP Over MPLS Network

Figure 8 illustrates DetNet enabled End Systems (hosts), connected to DetNet (DN) enabled IP networks, operating over a DetNet aware MPLS network. In this figure, Relay nodes sit at the boundary of the MPLS domain since the non-MPLS domain is DetNet aware. This figure is very similar to the DetNet MPLS Network figure in [I-D.ietf-detnet-dp-sol-mpls]. The primary difference is that the Relay nodes are at the edges of the MPLS domain and therefore function as T-PEs, and that service sub-layer functions are not provided over the DetNet IP network. The transit node functions show above are identical to those described in [I-D.ietf-detnet-dp-sol-mpls].

Figure 9 illustrates how relay nodes can provide service protection over an MPLS domain. In this case, CE1 and CE2 are IP DetNet end systems which are interconnected via a MPLS domain such as described in [I-D.ietf-detnet-dp-sol-mpls]. Note that R1 and R3 sit at the

edges of an MPLS domain and therefore are similar to T-PEs, while R2 sits in the middle of the domain and is therefore similar to an S-PE.

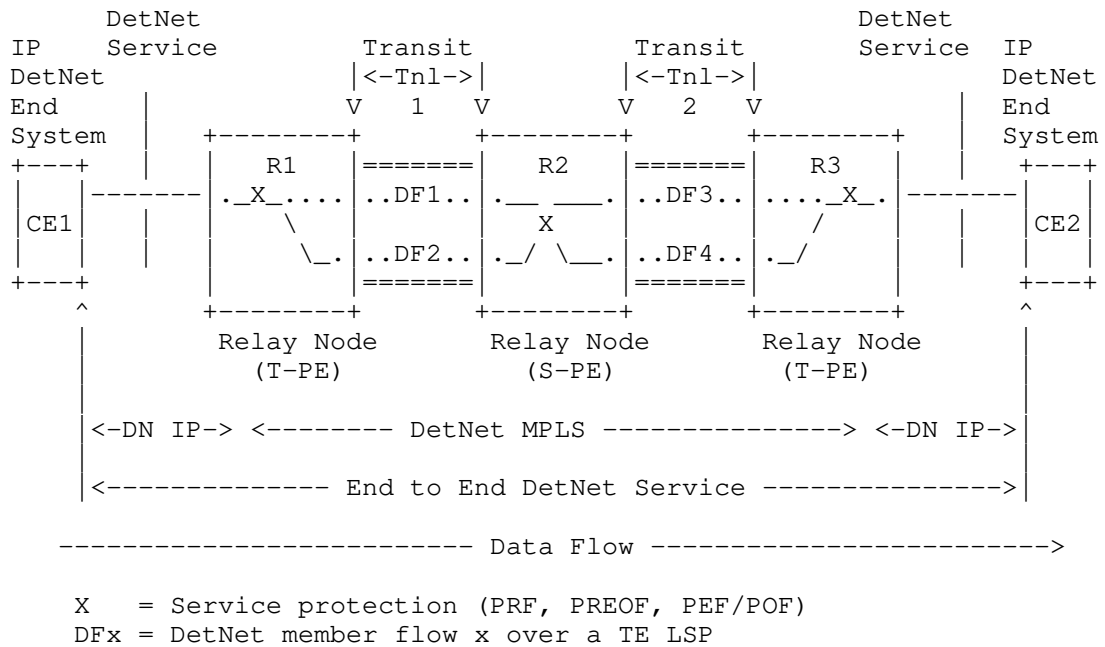


Figure 9: DetNet IP Over DetNet MPLS Network

[Editor's note: Text below in this sub-section is rather DetNet MPLS related, therefore candidate to be deleted in future versions.]

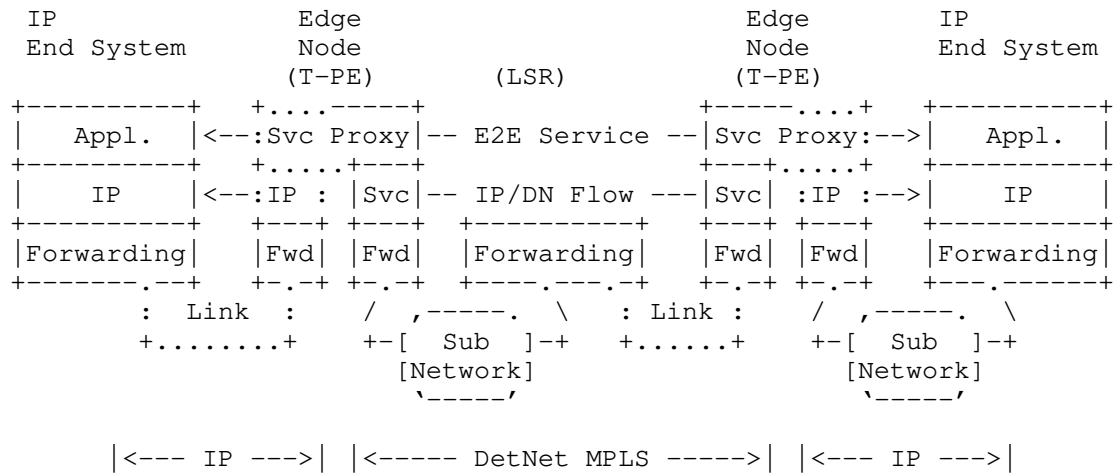
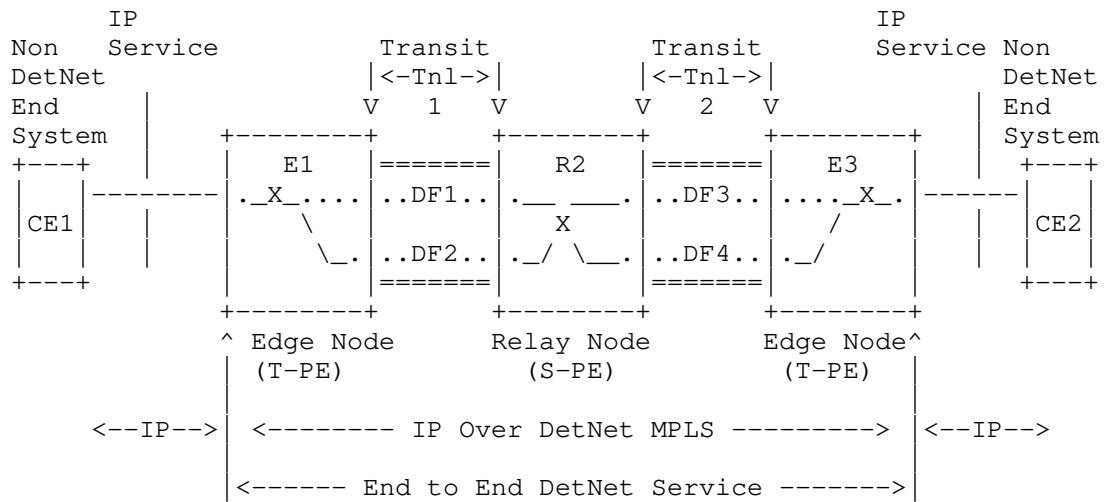


Figure 10: Non-DetNet Aware IP Over DetNet MPLS Network

Figure 10 illustrates non-DetNet enabled End Systems (hosts), connected to DetNet (DN) enabled MPLS network. It differs from Figure 8 in that the hosts and edge IP networks are not DetNet aware. In this case, edge nodes sit at the boundary of the MPLS domain since it is also a DetNet domain boundary. The edge nodes provide DetNet service proxies for the end applications by initiating and terminating DetNet service for the application's IP flows. While the node types differ, there is essentially no difference in data plane processing between relay and edges. There are likely to be differences in controller plane operation, particularly when distributed control plane protocols are used.

Figure 11 illustrates how it is still possible to provided DetNet service protection for non-DetNet aware end systems. This figures is basically the same as Figure 9, with the exception that CE1 and CE2 are non-DetNet aware end systems and E1 and E3 are edge nodes that replace the relay nodes R1 and R3.



X = Optional service protection (none, PRF, PREOF, PEF/POF)  
 DFx = DetNet member flow x over a TE LSP

Figure 11: MPLS-Based DetNet (non-MPLS End System)

[Editor's note: End of text being rather DetNet MPLS related.]

## 7.2. DetNet IP over DetNet MPLS Encapsulation

The basic encapsulation approach is to treat a DetNet IP flow as an app-flow from the DetNet MPLS app perspective. The corresponding example DetNet Sub-Network format is shown in Figure 12.

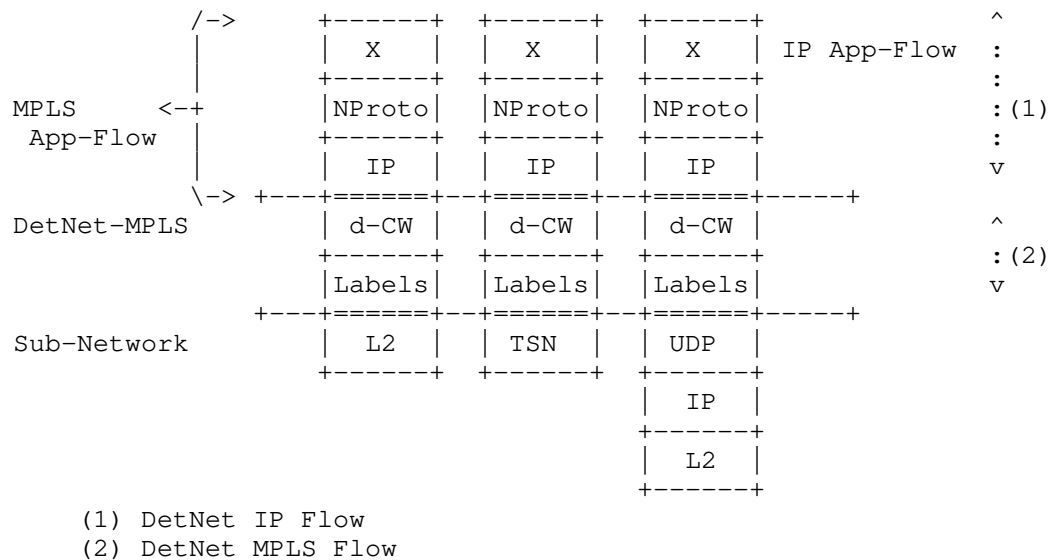


Figure 12: Example DetNet IP over MPLS Sub-Network Formats

In the figure, "IP App-Flow" indicates the payload carried by the DetNet IP data plane. "IP" and "NProto" indicate the fields described in Section 6.1.1 and Section 6.1.2, respectively. "MPLS App-Flow" indicates that an individual DetNet IP flow is the payload from the perspective of the DetNet MPLS data plane defined in [I-D.ietf-detnet-dp-sol-mpls].

Per [I-D.ietf-detnet-dp-sol-mpls], the DetNet MPLS data plane uses a single S-Label to support a single app flow. Section 6.1 states that a single DetNet flow is identified based on IP, and next level protocol, header information. It also defines that aggregation is supported (Section 6.4) through the use of prefixes, wildcards, bimbasks, and port ranges. Collectively, this results in the fairly straight forward procedures defined in this section.

As shown in Figure 2, DetNet relay nodes are responsible for the mapping of a DetNet flow, at the service sub-layer, from the IP to MPLS DetNet data planes and back again. Their related DetNet IP over DetNet MPLS data plane operation is comprised of two sets of procedures: the mapping of flow identifiers; and ensuring proper traffic treatment.

### 7.3. DetNet IP over DetNet MPLS Flow Identification Procedures

A relay node that sends a DetNet IP flows over a DetNet MPLS network MUST map a single DetNet IP flow into a single app-flow and MUST process that app-flow in accordance to the procedures defined in [I-D.ietf-detnet-dp-sol-mpls] Section 6.2. PRF MAY be supported for DetNet IP flows sent over an DetNet MPLS network. Aggregation as defined in Section 6.4 MAY be used to identify an individual DetNet IP flow. The provisioning of the mapping of DetNet IP flows to DetNet MPLS app-flow information MUST be supported via configuration, e.g., via the controller plane described in Section 5.

A relay node MAY be provisioned to handle packets received via the DetNet MPLS data plane as DetNet IP flows. A single incoming MPLS app-flow MAY be treated as a single DetNet IP flow, without examination of IP headers. Alternatively, packets received via the DetNet MPLS data plane MAY follow the normal DetNet IP flow identification procedures defined in Section 6.1. An implementation MUST support the provisioning of handling of received DetNet MPLS data plane as DetNet IP flows via configuration. Note that such configuration MAY include support from PEOF on the incoming DetNet MPLS flow.

### 7.4. DetNet IP over DetNet MPLS Traffic Treatment Procedures

The traffic treatment required for a particular DetNet IP flow is provisioned via configuration or the controller plane. When an DetNet IP flow is sent over DetNet MPLS a relay node MUST ensure that the provisioned DetNet IP traffic treatment is provided at the forwarding sub-layer as described in [I-D.ietf-detnet-dp-sol-mpls] Section 6.2. Note that the PRF function can also be used when sending over MPLS.

Traffic treatment for DetNet IP flows received over the DetNet MPLS data plane MUST follow Section 6.3.

## 8. Mapping DetNet IP Flows to IEEE 802.1 TSN

[Authors note: how do we handle control protocols such as ICMP, IPsec, etc.]

This section covers how DetNet IP flows operate over an IEEE 802.1 TSN sub-network. Figure 13 illustrates such a scenario, where two IP (DetNet) nodes are interconnected by a TSN sub-network. Node-1 is single homed and Node-2 is dual-homed. IP nodes can be (1) DetNet IP End System, (2) DetNet IP Edge or Relay node or (3) IP End System.

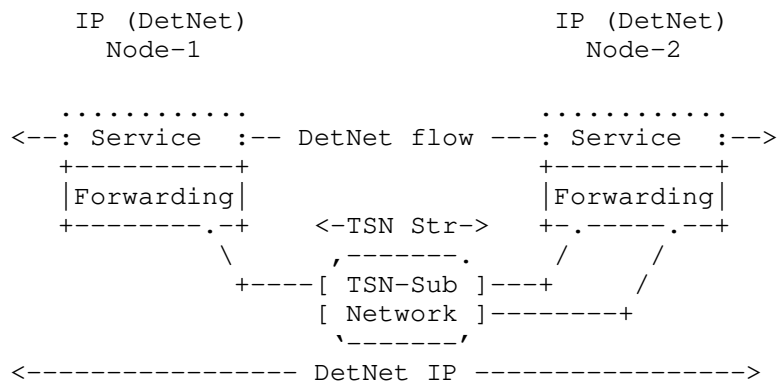


Figure 13: DetNet (DN) Enabled IP Network over a TSN sub-network

The Time-Sensitive Networking (TSN) Task Group of the IEEE 802.1 Working Group have defined (and are defining) a number of amendments to IEEE 802.1Q [IEEE8021Q] that provide zero congestion loss and bounded latency in bridged networks. Furthermore IEEE 802.1CB [IEEE8021CB] defines frame replication and elimination functions for reliability that should prove both compatible with and useful to, DetNet networks. All these functions have to identify flows those require TSN treatment.

As is the case for DetNet, a Layer 2 network node such as a bridge may need to identify the specific DetNet flow to which a packet belongs in order to provide the TSN/DetNet QoS for that packet. It also may need additional marking, such as the priority field of an IEEE Std 802.1Q VLAN tag, to give the packet proper service.

TSN capabilities of the TSN sub-network are made available for IP (DetNet) flows via the protocol interworking function defined in IEEE 802.1CB [IEEE8021CB]. For example, applied on the TSN edge port connected to the IP (DetNet) node it can convert an ingress unicast IP (DetNet) flow to use a specific multicast destination MAC address and VLAN, in order to direct the packet through a specific path inside the bridged network. A similar interworking pair at the other end of the TSN sub-network would restore the packet to its original destination MAC address and VLAN.

Placement of TSN functions depends on the TSN capabilities of nodes. IP (DetNet) Nodes may or may not support TSN functions. For a given TSN Stream (i.e., DetNet flow) an IP (DetNet) node is treated as a Talker or a Listener inside the TSN sub-network.



### 8.1. TSN Stream ID Mapping

DetNet IP Flow and TSN Stream mapping is based on the active Stream Identification function, that operates at the frame level. IEEE 802.1CB [IEEE8021CB] defines an Active Destination MAC and VLAN Stream identification function, what can replace some Ethernet header fields namely (1) the destination MAC-address, (2) the VLAN-ID and (3) priority parameters with alternate values. Replacement is provided for the frame passed down the stack from the upper layers or up the stack from the lower layers.

Active Destination MAC and VLAN Stream identification can be used within a Talker to set flow identity or a Listener to recover the original addressing information. It can be used also in a TSN bridge that is providing translation as a proxy service for an End System. As a result IP (DetNet) flows can be mapped to use a particular {MAC-address, VLAN} pair to match the Stream in the TSN sub-network.

[Editor's note: there are no requirement on IP DetNet nodes in case of "IP (DetNet) node without TSN functions" scenarios. Paragraph and figure below are candidates to be deleted in future versions.]

From the TSN sub-network perspective DetNet IP nodes without any TSN functions can be treated as TSN-unaware Talker or Listener. In such cases relay nodes in the TSN sub-network MUST modify the Ethernet encapsulation of the DetNet IP flow (e.g., MAC translation, VLAN-ID setting, Sequence number addition, etc.) to allow proper TSN specific handling of the flow inside the sub-network. This is illustrated in Figure 14.

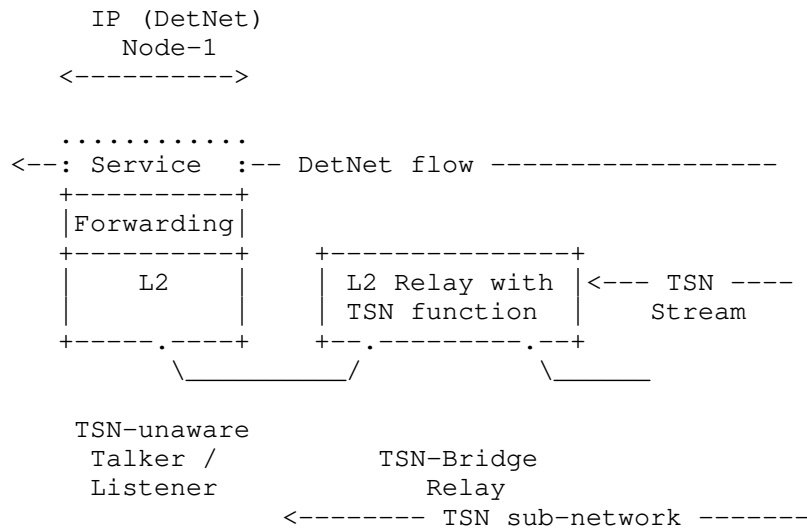


Figure 14: IP (DetNet) node without TSN functions

IP (DetNet) nodes being TSN-aware can be treated as a combination of a TSN-unaware Talker/Listener and a TSN-Relay, as shown in Figure 15. In such cases the IP (DetNet) node MUST provide the TSN sub-network specific Ethernet encapsulation over the link(s) towards the sub-network. An TSN-aware IP (DetNet) node MUST support the following TSN components:

1. For recognizing flows:
  - \* Stream Identification
2. For FRER used inside the TSN domain, additionally:
  - \* Sequencing function
  - \* Sequence encode/decode function
3. For FRER when the node is a replication or elimination point, additionally:
  - \* Stream splitting function
  - \* Individual recovery function

[Editor's note: Should we added here requirements regarding IEEE 802.1Q C-VLAN component?]

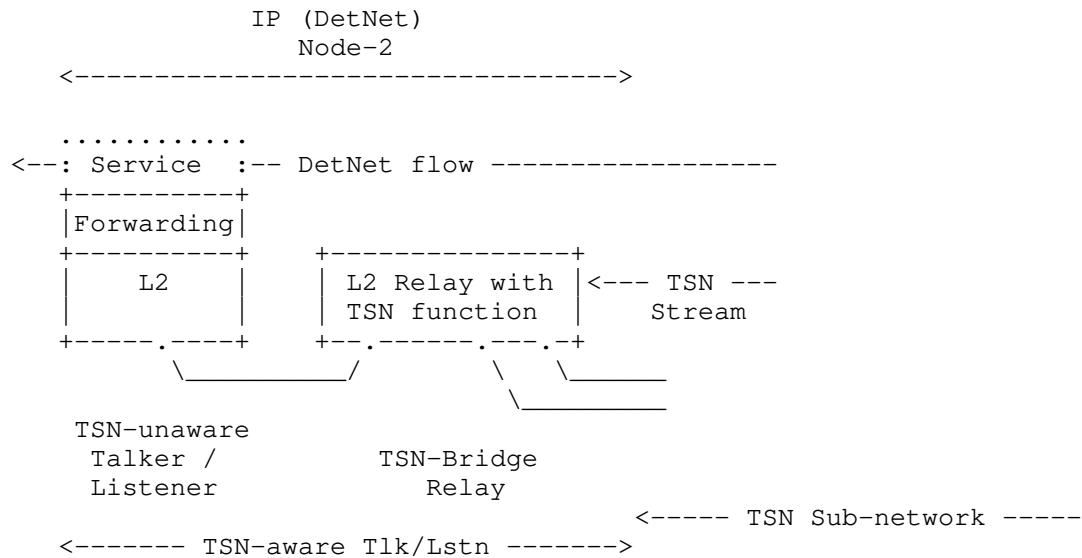


Figure 15: IP (DetNet) node with TSN functions

A Stream identification component MUST be able to instantiate the following functions (1) Active Destination MAC and VLAN Stream identification function, (2) IP Stream identification function and (3) the related managed objects in Clause 9 of IEEE 802.1CB [IEEE8021CB]. IP Stream identification function provides a 6-tuple match.

The Sequence encode/decode function MUST support the Redundancy tag (R-TAG) format as per Clause 7.8 of IEEE 802.1CB [IEEE8021CB].

## 8.2. TSN Usage of FRER

TSN Streams supporting DetNet flows may use Frame Replication and Elimination for Redundancy (FRER) [802.1CB] based on the loss service requirements of the TSN Stream, which is derived from the DetNet service requirements of the DetNet mapped flow. The specific operation of FRER is not modified by the use of DetNet and follows IEEE 802.1CB [IEEE8021CB].

FRER function and the provided service recovery is available only within the TSN sub-network (as shown in Figure 6) as the Stream-ID and the TSN sequence number are not valid outside the sub-network. An IP (DetNet) node represents a L3 border and as such it terminates all related information elements encoded in the L2 frames.

### 8.3. Procedures

[Editor's note: This section is TBD - covers required behavior of a TSN-aware DetNet node using a TSN underlay.]

This section provides DetNet IP data plane procedures to interwork with a TSN underlay sub-network when the IP (DetNet) node acts as a TSN-aware Talker or Listener (see Figure 15). These procedures have been divided into the following areas: flow identification, mapping of a DetNet flow to a TSN Stream and ensure proper TSN encapsulation.

Flow identification procedures are described in Section 6.1. A TSN-aware IP (DetNet) node SHALL support the Stream Identification TSN components as per IEEE 802.1CB [IEEE8021CB].

Implementations of this document SHALL use management and control information to map a DetNet flow to a TSN Stream. N:1 mapping (aggregating DetNet flows in a single TSN Stream) SHALL be supported. The management or control function that provisions flow mapping SHALL ensure that adequate resources are allocated and configured to provide proper service requirements of the mapped flows.

For proper TSN encapsulation implementations of this document SHALL support active Stream Identification function as defined in chapter 6.6 in IEEE 802.1CB [IEEE8021CB].

A TSN-aware IP (DetNet) node SHALL support Ethernet encapsulation with Redundancy tag (R-TAG) as per chapter 7.8 in IEEE 802.1CB [IEEE8021CB].

Depending whether FRER functions are used in the TSN sub-network to serve the mapped TSN Stream, a TSN-aware IP (DetNet) node SHALL support Sequencing function and Sequence encode/decode function as per chapter 7.4 and 7.6 in IEEE 802.1CB [IEEE8021CB]. Furthermore when a TSN-aware IP (DetNet) node acting as a replication or elimination point for FRER it SHALL implement the Stream splitting function and the Individual recovery function as per chapter 7.7 and 7.5 in IEEE 802.1CB [IEEE8021CB].

### 8.4. Management and Control Implications

[Editor's note: This section is TBD Covers Creation, mapping, removal of TSN Stream IDs, related parameters and, when needed, configuration of FRER. Supported by management/control plane.]

DetNet flow and TSN Stream mapping related information are required only for TSN-aware IP (DetNet) nodes. From the Data Plane

perspective there is no practical difference based on the origin of flow mapping related information (management plane or control plane).

TSN-aware DetNet IP nodes are member of both the DetNet domain and the TSN sub-network. Within the TSN sub-network the TSN-aware IP (DetNet) node has a TSM-aware Talker/Listener role, so TSN specific management and control plane functionalities must be implemented. There are many similarities in the management plane techniques used in DetNet and TSN, but that is not the case for the control plane protocols. For example, RSVP-TE and MSRP behaves differently. Therefore management and control plane design is an important aspect of scenarios, where mapping between DetNet and TSN is required.

In order to use a TSN sub-network between DetNet nodes, DetNet specific information must be converted to TSN sub-network specific ones. DetNet flow ID and flow related parameters/requirements must be converted to a TSN Stream ID and stream related parameters/requirements. Note that, as the TSN sub-network is just a portion of the end2end DetNet path (i.e., single hop from IP perspective), some parameters (e.g., delay) may differ significantly. Other parameters (like bandwidth) also may have to be tuned due to the L2 encapsulation used in the TSN sub-network.

In some case it may be challenging to determine some TSN Stream related information. For example on a TSN-aware IP (DetNet) node that acts as a Talker, it is quite obvious which DetNet node is the Listener of the mapped TSN stream (i.e., the IP Next-Hop). However it may be not trivial to locate the point/interface where that Listener is connected to the TSN sub-network. Such attributes may require interaction between control and management plane functions and between DetNet and TSN domains.

Mapping between DetNet flow identifiers and TSN Stream identifiers, if not provided explicitly, can be done by a TSN-aware IP (DetNet) node locally based on information provided for configuration of the TSN Stream identification functions (IP Stream identification and active Stream identification function).

Triggering the setup/modification of a TSN Stream in the TSN sub-network is an example where management and/or control plane interactions are required between the DetNet and TSN sub-network. TSN-unaware IP (DetNet) nodes make such a triggering even more complicated as they are fully unaware of the sub-network and run independently.

Configuration of TSN specific functions (e.g., FRER) inside the TSN sub-network is a TSN specific decision and may not be visible in the DetNet domain.

## 9. Security Considerations

The security considerations of DetNet in general are discussed in [I-D.ietf-detnet-architecture] and [I-D.ietf-detnet-security]. Other security considerations will be added in a future version of this draft.

## 10. IANA Considerations

TBD.

## 11. Contributors

RFC7322 limits the number of authors listed on the front page of a draft to a maximum of 5, far fewer than the 20 individuals below who made important contributions to this draft. The editor wishes to thank and acknowledge each of the following authors for contributing text to this draft. See also Section 12.

Loa Andersson  
Huawei  
Email: loa@pi.nu

Yuanlong Jiang  
Huawei  
Email: jiangyuanlong@huawei.com

Norman Finn  
Huawei  
3101 Rio Way  
Spring Valley, CA 91977  
USA  
Email: norman.finn@mail01.huawei.com

Janos Farkas  
Ericsson  
Magyar Tudosok krt. 11  
Budapest 1117  
Hungary  
Email: janos.farkas@ericsson.com

Carlos J. Bernardos  
Universidad Carlos III de Madrid  
Av. Universidad, 30  
Leganes, Madrid 28911  
Spain  
Email: cjbc@it.uc3m.es

Tal Mizrahi  
Marvell  
6 Hamada st.  
Yokneam  
Israel  
Email: talmi@marvell.com

Lou Berger  
LabN Consulting, L.L.C.  
Email: lberger@labn.net

Andrew G. Malis  
Huawei Technologies  
Email: agmalis@gmail.com

## 12. Acknowledgements

The author(s) ACK and NACK.

The following people were part of the DetNet Data Plane Solution Design Team:

Jouni Korhonen

Janos Farkas

Norman Finn

Balazs Varga

Loa Andersson

Tal Mizrahi

David Mozes

Yuanlong Jiang

Andrew Malis

Carlos J. Bernardos

The DetNet chairs serving during the DetNet Data Plane Solution Design Team:

Lou Berger

Pat Thaler

Thanks for Stewart Bryant for his extensive review of the previous versions of the document.

## 13. References

### 13.1. Normative references

- [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, DOI 10.17487/RFC0768, August 1980, <<https://www.rfc-editor.org/info/rfc768>>.
- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.



- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <<https://www.rfc-editor.org/info/rfc1812>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2211] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", RFC 2211, DOI 10.17487/RFC2211, September 1997, <<https://www.rfc-editor.org/info/rfc2211>>.
- [RFC2212] Shenker, S., Partridge, C., and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, DOI 10.17487/RFC2212, September 1997, <<https://www.rfc-editor.org/info/rfc2212>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <<https://www.rfc-editor.org/info/rfc3270>>.

- [RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, DOI 10.17487/RFC3473, January 2003, <<https://www.rfc-editor.org/info/rfc3473>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4303] Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.
- [RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <<https://www.rfc-editor.org/info/rfc5462>>.
- [RFC6003] Papadimitriou, D., "Ethernet Traffic Parameters", RFC 6003, DOI 10.17487/RFC6003, October 2010, <<https://www.rfc-editor.org/info/rfc6003>>.
- [RFC7608] Boucadair, M., Petrescu, A., and F. Baker, "IPv6 Prefix Length Recommendation for Forwarding", BCP 198, RFC 7608, DOI 10.17487/RFC7608, July 2015, <<https://www.rfc-editor.org/info/rfc7608>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.

### 13.2. Informative references

- [G.8275.1] International Telecommunication Union, "Precision time protocol telecom profile for phase/time synchronization with full timing support from the network", ITU-T G.8275.1/Y.1369.1 G.8275.1, June 2016, <<https://www.itu.int/rec/T-REC-G.8275.1/en>>.

- [G.8275.2]  
International Telecommunication Union, "Precision time protocol telecom profile for phase/time synchronization with partial timing support from the network", ITU-T G.8275.2/Y.1369.2 G.8275.2, June 2016, <<https://www.itu.int/rec/T-REC-G.8275.2/en>>.
- [I-D.ietf-detnet-architecture]  
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-11 (work in progress), February 2019.
- [I-D.ietf-detnet-dp-sol-mpls]  
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane Encapsulation", draft-ietf-detnet-dp-sol-mpls-01 (work in progress), October 2018.
- [I-D.ietf-detnet-flow-information-model]  
Farkas, J., Varga, B., Cummings, R., and Y. Jiang, "DetNet Flow Information Model", draft-ietf-detnet-flow-information-model-03 (work in progress), March 2019.
- [I-D.ietf-detnet-security]  
Mizrahi, T., Grossman, E., Hacker, A., Das, S., Dowdell, J., Austad, H., Stanton, K., and N. Finn, "Deterministic Networking (DetNet) Security Considerations", draft-ietf-detnet-security-04 (work in progress), March 2019.
- [I-D.ietf-teas-pce-native-ip]  
Wang, A., Zhao, Q., Khasanov, B., Chen, H., and R. Mallya, "PCE in Native IP Network", draft-ietf-teas-pce-native-ip-02 (work in progress), October 2018.
- [IEEE1588]  
IEEE, "IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2", 2008.
- [IEEE8021CB]  
Finn, N., "Draft Standard for Local and metropolitan area networks - Seamless Redundancy", IEEE P802.1CB /D2.1 P802.1CB, December 2015, <<http://www.ieee802.org/1/files/private/cb-drafts/d2/802-1CB-d2-1.pdf>>.

- [IEEE8021Q] IEEE 802.1, "Standard for Local and metropolitan area networks--Bridges and Bridged Networks (IEEE Std 802.1Q-2014)", 2014, <<http://standards.ieee.org/about/get/>>.
- [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <<https://www.rfc-editor.org/info/rfc1122>>.
- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <<https://www.rfc-editor.org/info/rfc2205>>.
- [RFC2386] Crawley, E., Nair, R., Rajagopalan, B., and H. Sandick, "A Framework for QoS-based Routing in the Internet", RFC 2386, DOI 10.17487/RFC2386, August 1998, <<https://www.rfc-editor.org/info/rfc2386>>.
- [RFC3670] Moore, B., Durham, D., Strassner, J., Westerinen, A., and W. Weiss, "Information Model for Describing Network Device QoS Datapath Mechanisms", RFC 3670, DOI 10.17487/RFC3670, January 2004, <<https://www.rfc-editor.org/info/rfc3670>>.
- [RFC5575] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and D. McPherson, "Dissemination of Flow Specification Rules", RFC 5575, DOI 10.17487/RFC5575, August 2009, <<https://www.rfc-editor.org/info/rfc5575>>.
- [RFC5654] Niven-Jenkins, B., Ed., Brungard, D., Ed., Betts, M., Ed., Sprecher, N., and S. Ueno, "Requirements of an MPLS Transport Profile", RFC 5654, DOI 10.17487/RFC5654, September 2009, <<https://www.rfc-editor.org/info/rfc5654>>.
- [RFC5777] Korhonen, J., Tschofenig, H., Arumaithurai, M., Jones, M., Ed., and A. Lior, "Traffic Classification and Quality of Service (QoS) Attributes for Diameter", RFC 5777, DOI 10.17487/RFC5777, February 2010, <<https://www.rfc-editor.org/info/rfc5777>>.
- [RFC6434] Jankiewicz, E., Loughney, J., and T. Narten, "IPv6 Node Requirements", RFC 6434, DOI 10.17487/RFC6434, December 2011, <<https://www.rfc-editor.org/info/rfc6434>>.

- [RFC7551] Zhang, F., Ed., Jing, R., and R. Gandhi, Ed., "RSVP-TE Extensions for Associated Bidirectional Label Switched Paths (LSPs)", RFC 7551, DOI 10.17487/RFC7551, May 2015, <<https://www.rfc-editor.org/info/rfc7551>>.
- [RFC7657] Black, D., Ed. and P. Jones, "Differentiated Services (Diffserv) and Real-Time Communication", RFC 7657, DOI 10.17487/RFC7657, November 2015, <<https://www.rfc-editor.org/info/rfc7657>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8169] Mirsky, G., Ruffini, S., Gray, E., Drake, J., Bryant, S., and A. Vainshtein, "Residence Time Measurement in MPLS Networks", RFC 8169, DOI 10.17487/RFC8169, May 2017, <<https://www.rfc-editor.org/info/rfc8169>>.
- [RFC8283] Farrel, A., Ed., Zhao, Q., Ed., Li, Z., and C. Zhou, "An 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, <<https://www.rfc-editor.org/info/rfc8283>>.

#### Appendix A. Example of DetNet Data Plane Operation

[Editor's note: Add a simplified example of DetNet data plane and how labels etc work in the case of MPLS-based PSN and utilizing PREOF. The figure is subject to change depending on the further DT decisions on the label handling..]

#### Appendix B. Example of Pinned Paths Using IPv6

TBD.

#### Authors' Addresses

Jouni Korhonen (editor)

Email: [jouni.nospam@gmail.com](mailto:jouni.nospam@gmail.com)

Balazs Varga (editor)  
Ericsson  
Magyar Tudosok krt. 11.  
Budapest 1117  
Hungary

Email: [balazs.a.varga@ericsson.com](mailto:balazs.a.varga@ericsson.com)

DetNet  
Internet-Draft  
Intended status: Standards Track  
Expires: September 11, 2019

J. Korhonen, Ed.  
B. Varga, Ed.  
Ericsson  
March 10, 2019

DetNet MPLS Data Plane Encapsulation  
draft-ietf-detnet-dp-sol-mpls-02

Abstract

This document specifies the Deterministic Networking data plane when operating over an MPLS Packet Switched 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 September 11, 2019.

Copyright Notice

Copyright (c) 2019 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 . . . . .	3
2. Terminology . . . . .	4
2.1. Terms Used in This Document . . . . .	4
2.2. Abbreviations . . . . .	4
3. Requirements Language . . . . .	5
4. DetNet MPLS Data Plane Overview . . . . .	6
4.1. Layers of DetNet Data Plane . . . . .	6
4.2. DetNet MPLS Data Plane Scenarios . . . . .	7
4.2.1. IP Over DetNet MPLS Data Plane Scenarios . . . . .	9
4.2.2. IEEE 802.1 TSN Over DetNet MPLS Data Plane Scenario . . . . .	12
4.3. Packet Flow Example with Service Protection . . . . .	14
5. DetNet MPLS Data Plane Considerations . . . . .	15
5.1. End-System Specific Considerations . . . . .	16
5.2. Sub-Network Considerations . . . . .	17
6. MPLS-Based DetNet Data Plane Solution . . . . .	18
6.1. DetNet Over MPLS Encapsulation Components . . . . .	18
6.2. MPLS Data Plane Encapsulation . . . . .	19
6.2.1. DetNet Control Word and the DetNet Sequence Number . . . . .	20
6.2.2. S-Labels . . . . .	21
6.2.3. F-Labels . . . . .	24
6.3. OAM Indication . . . . .	26
6.4. Flow Aggregation . . . . .	27
6.4.1. Aggregation at the LSP . . . . .	28
6.4.2. Aggregating DetNet Flows as a new DetNet flow . . . . .	28
6.4.3. Simple Aggregation at the DetNet Layer . . . . .	29
6.5. Service Sub-Layer Considerations . . . . .	29
6.5.1. Edge Node Processing . . . . .	30
6.5.2. Relay Node Processing . . . . .	31
6.6. Forwarding Sub-Layer Considerations . . . . .	31
6.6.1. Class of Service . . . . .	31
6.6.2. Quality of Service . . . . .	32
6.6.3. Cross-DetNet Flow Resource Aggregation . . . . .	32
6.6.4. Layer 2 Addressing and QoS Considerations . . . . .	33
6.6.5. Time Synchronization . . . . .	34
7. Controller Plane (Management and Control) Considerations . . . . .	34
7.1. S-Label and F-Label Assignment and Distribution . . . . .	35
7.2. Packet Replication, Elimination, and Ordering (PREOF) . . . . .	36
7.3. Contention Loss and Jitter Reduction . . . . .	36
7.4. Bidirectional Traffic . . . . .	37
7.5. Flow Aggregation Control . . . . .	38
7.6. DetNet Controller (Control and Management) Plane Requirements . . . . .	38
8. DetNet MPLS Operation Over IEEE 802.1 TSN Sub-Networks . . . . .	39
8.1. Mapping of TSN Stream ID and Sequence Number . . . . .	41
8.2. TSN Usage of FRER . . . . .	42



8.3. Management and Control Implications . . . . .	42
9. DetNet MPLS Operation over DetNet	
IP PSNs . . . . .	43
10. Security Considerations . . . . .	45
11. IANA Considerations . . . . .	45
12. Contributors . . . . .	45
13. Acknowledgements . . . . .	46
14. References . . . . .	47
14.1. Normative References . . . . .	47
14.2. Informative References . . . . .	49
Appendix A. Example of DetNet Data Plane Operation . . . . .	52
Authors' Addresses . . . . .	52

## 1. Introduction

Deterministic Networking (DetNet) is a service that can be offered by a network to DetNet flows. DetNet provides these flows with a low packet loss rates and assured maximum end-to-end delivery latency. General background and concepts of DetNet can be found in [I-D.ietf-detnet-architecture].

The DetNet Architecture decomposes the DetNet related data plane functions into two sub-layers: a service sub-layer and a forwarding sub-layer. The service sub-layer is used to provide DetNet service protection and reordering. The forwarding sub-layer is used to provides congestion protection (low loss, assured latency, and limited reordering) leveraging MPLS Traffic Engineering mechanisms.

This document specifies the DetNet data plane operation and the on-wire encapsulation of DetNet flows over an MPLS-based Packet Switched Network (PSN). The specified encapsulation provides the building blocks to enable the DetNet service and forwarding sub-layer functions and supports flow identification as described in the DetNet Architecture. As part of the service sub-layer functions, this document describes DetNet node data plane operation. It also describes the function and operation of the Packet Replication (PRF) Packet Elimination (PEF) and Packet Ordering (POF) functions with an MPLS data plane. It also describes an MPLS-based DetNet forwarding sub-layer that eliminates (or reduces) contention loss and provides bounded latency for DetNet flows.

MPLS encapsulated DetNet flows can be carried over network technologies that can provide the DetNet required level of service. This document defines examples of such, specifically carrying DetNet MPLS flows over IEEE 802.1 TSN sub-networks, and over DetNet IP PSN.

The intent is for this document to support different traffic types being mapped over DetNet MPLS, but this is out side the scope of this

document. An example of such can be found in [I-D.ietf-detnet-dp-sol-ip]. This document also allows for, but does not define, associated controller plane and Operations, Administration, and Maintenance (OAM) functions.

## 2. Terminology

### 2.1. Terms Used in This Document

This document uses the terminology established in the DetNet architecture [I-D.ietf-detnet-architecture], and the reader is assumed to be familiar with that document and its terminology.

The following terminology is introduced in this document:

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.
d-CW	A DetNet Control Word (d-CW) is used for sequencing and identifying duplicate packets of a DetNet flow at the DetNet service sub-layer.

### 2.2. Abbreviations

The following abbreviations are used in this document:

AC	Attachment Circuit.
CE	Customer Edge equipment.
CoS	Class of Service.
CW	Control Word.
DetNet	Deterministic Networking.
DF	DetNet Flow.
DN-IWF	DetNet Inter-Working Function.
L2	Layer 2.

L2VPN	Layer 2 Virtual Private Network.
L3	Layer 3.
LSR	Label Switching Router.
MPLS	Multiprotocol Label Switching.
MPLS-TE	Multiprotocol Label Switching - Traffic Engineering.
MPLS-TP	Multiprotocol Label Switching - Transport Profile.
MS-PW	Multi-Segment PseudoWire (MS-PW).
NSP	Native Service Processing.
OAM	Operations, Administration, and Maintenance.
PE	Provider Edge.
PEF	Packet Elimination Function.
PRF	Packet Replication Function.
PREOF	Packet Replication, Elimination and Ordering Functions.
POF	Packet Ordering Function.
PSN	Packet Switched Network.
PW	PseudoWire.
QoS	Quality of Service.
S-PE	Switching Provider Edge.
T-PE	Terminating Provider Edge.
TSN	Time-Sensitive Network.

### 3. 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.

## 4. DetNet MPLS Data Plane Overview

### 4.1. Layers of DetNet Data Plane

This document describes how DetNet flows are carried over MPLS networks. The DetNet Architecture, [I-D.ietf-detnet-architecture], decomposes the DetNet data plane into two sub-layers: a service sub-layer and a forwarding sub-layer. The basic approach defined in this document supports the DetNet service sub-layer based on existing pseudowire (PW) encapsulations and mechanisms, and supports the DetNet forwarding sub-layer based on existing MPLS Traffic Engineering encapsulations and mechanisms. Background on PWs can be found in [RFC3985] and [RFC3031]. Background on MPLS Traffic Engineering can be found in [RFC3272] and [RFC3209].

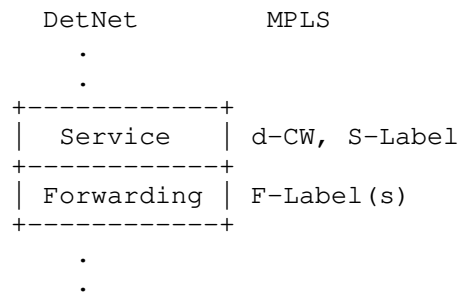


Figure 1: DetNet Adaptation to MPLS Data Plane

The DetNet MPLS data plane approach defined in this document is shown in Figure 1. The service sub-layer is supported by a DetNet control word (d-CW) which conforms to the Generic PW MPLS Control Word (PWMCW) defined in [RFC4385]. A d-CW identifying service label (S-Label) is also used.

A node operating on a DetNet flow in the Detnet service sub-layer, i.e. a node processing a DetNet packet which has the S-Label as top of stack uses the local context associated with that S-Label, for example a received F-Label, to determine what local DetNet operation(s) are applied to that packet. An S-Label may be unique when taken from the platform label space [RFC3031], which would enable correct DetNet flow identification regardless of which input interface or LSP the packet arrives on.

The DetNet MPLS data plane builds on MPLS Traffic Engineering encapsulations and mechanisms to provide a forwarding sub-layer that is responsible for providing resource allocation and explicit routes. The forwarding sub-layer is supported by one or more forwarding labels (F-Labels).

## 4.2. DetNet MPLS Data Plane Scenarios

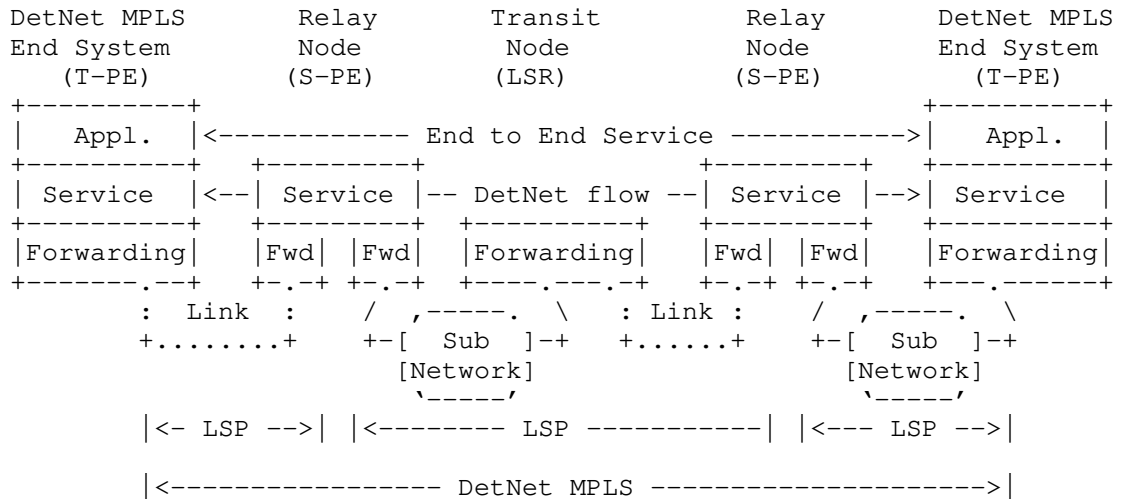


Figure 2: A DetNet MPLS Network

Figure 2 illustrates a hypothetical DetNet MPLS-only network composed of DetNet aware MPLS enabled end systems, operating over a DetNet aware MPLS network. In this figure, relay nodes sit at MPLS LSP boundaries and transit nodes are LSRs.

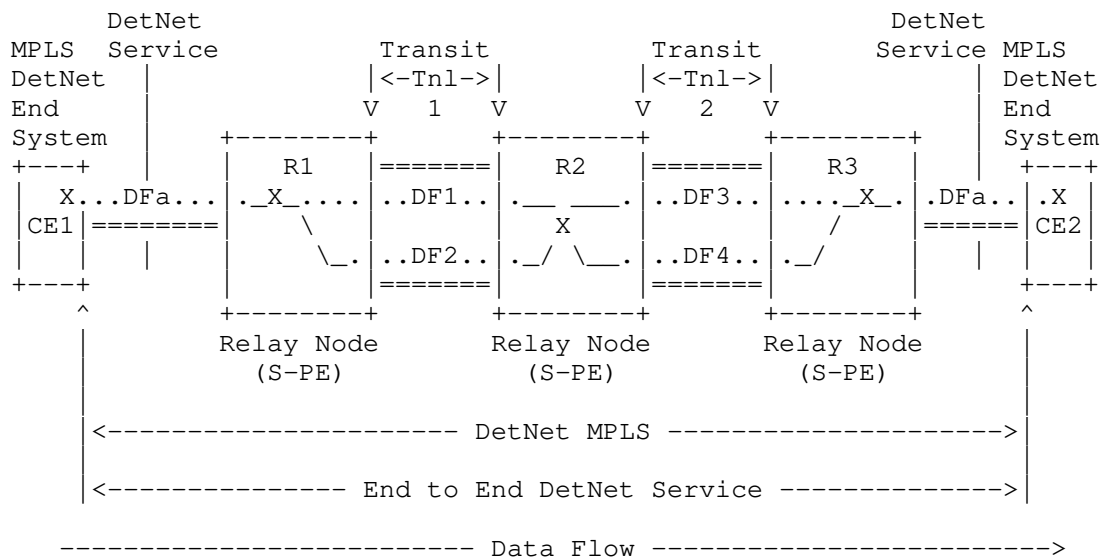
DetNet end system and relay nodes are DetNet service sub-layer aware, understand the particular needs of DetNet flows and provide both DetNet service and forwarding sub-layer functions. They add, remove and process d-CWs, S-Labels and F-labels as needed. MPLS enabled end system and relay nodes can enhance the reliability of delivery by enabling the replication of packets where multiple copies, possibly over multiple paths, are forwarded through the DetNet domain. They can also eliminate surplus previously replicated copies of DetNet packets. DetNet MPLS nodes provide functionality similar to T-PEs when they sit at the edge of an MPLS domain, and functionality similar to S-PEs when they are in the middle of an MPLS domain, see [RFC6073]. End system and relay nodes also include DetNet forwarding sub-layer functions, support for notably explicit routes, and resources allocation to eliminate (or reduce) congestion loss and jitter.

DetNet transit nodes reside wholly within a DetNet domain, and also provide DetNet forwarding sub-layer functions in accordance with the performance required by a DetNet flow carried over an LSP. Unlike other DetNet node types, transit nodes provide no service sub-layer

processing. In a DetNet MPLS network, transit nodes may be DetNet service aware or may be DetNet unaware MPLS Label Switching Routers (LSRs). In this latter case, such LSRs would be unaware of the special requirements of the DetNet service sub-layer, but would still provide traffic engineering services and the QoS need to ensure that the (TE) LSPs meet the service requirements of the carried DetNet flows.

The LSPs may be provided by any MPLS controller method. For example they may be provisioned via a management plane, RSVP-TE, MPLS-TP, or MPLS Segment Routing (when extended to support resource allocation).

Figure 3 illustrates how an end to end MPLS-based DetNet service is provided in a more detail. In this figure, the end systems, CE1 and CE2, are able to send and receive MPLS encapsulated DetNet flows, and R1, R2 and R3 are relay nodes as they sit in the middle of a DetNet network. The 'X' in the end systems, and relay nodes represents potential DetNet compound flow packet replication and elimination points. In this example, service protection is supported over four DetNet member flows and TE LSPs. For a unidirectional flow, R1 supports PRF, R2 supports PREOF and R3 supports PEF and POF. Note that the relay nodes may change the underlying forwarding sub-layer, for example tunneling MPLS over IEEE 802.1 TSN Section 8, or simply over interconnect network links.



X = Optional service protection (none, PRF, PREOF, PEF/POF)  
DFx = DetNet member flow x over a TE LSP

Figure 3: MPLS-Based Native DetNet

As previously mentioned, this document specifies how MPLS is used to support DetNet flows using an MPLS data plane as well as how such can be mapped to IEEE 802.1 TSN and IP DetNet PSNs. An equally important scenario is when IP is supported over DetNet MPLS and this is covered in [I-D.ietf-detnet-dp-sol-ip]. Another important scenario is where an Ethernet Layer 2 service is supported over DetNet MPLS and this is covered in [TBD-TSN-OVER-DETNET].

#### 4.2.1. IP Over DetNet MPLS Data Plane Scenarios

[Author's note: this section to be moved to IP sol draft]

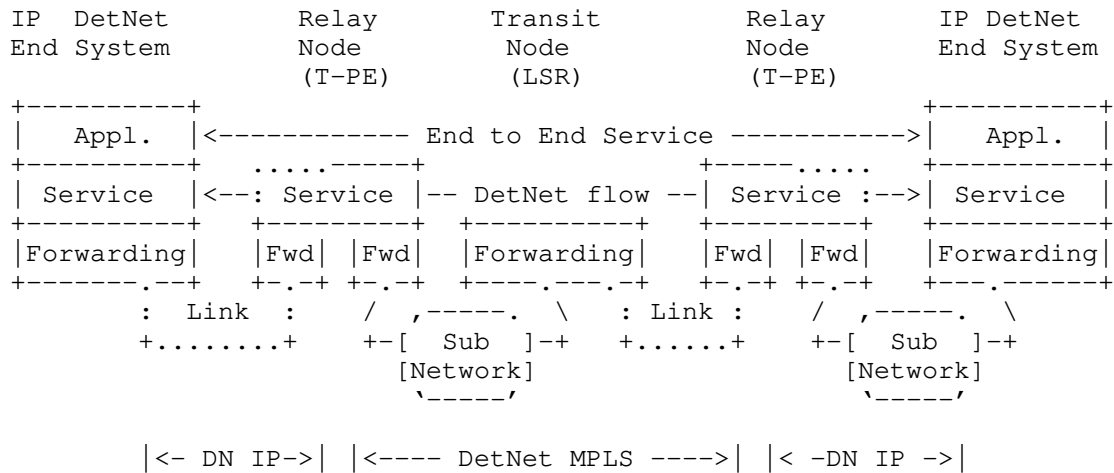
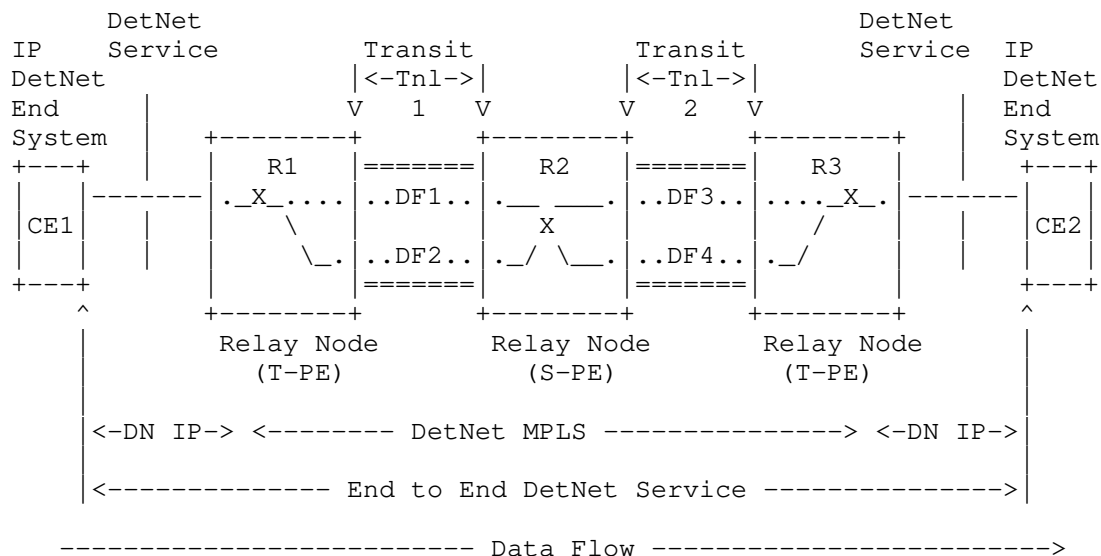


Figure 4: DetNet IP Over MPLS Network

Figure 4 illustrates DetNet enabled End Systems (hosts), connected to DetNet (DN) enabled IP networks, operating over a DetNet aware MPLS network. In this figure, Relay nodes sit at the boundary of the MPLS domain since the non-MPLS domain is DetNet aware. This figure is very similar to Figure 2. The primary difference is that the Relay nodes are at the edges of the MPLS domain and therefore function as T-PEs, and that service sub-layer functions are not provided over the DetNet IP network. There is no difference in transit node function.

Figure 5 illustrates how relay nodes can provide service protection over the MPLS domain. In this case, CE1 and CE2 are IP DetNet end systems which are interconnected via a MPLS domain such as previously shown in Figure 3. Note that R1 and R3 sit at the edges of an MPLS domain and therefore are similar to T-PEs, while R2 sits in the middle of the domain and is therefore similar to an S-PE.





X = Service protection (PRF, PREOF, PEF/POF)  
DFx = DetNet member flow x over a TE LSP

Figure 5: DetNet IP Over DetNet MPLS Network

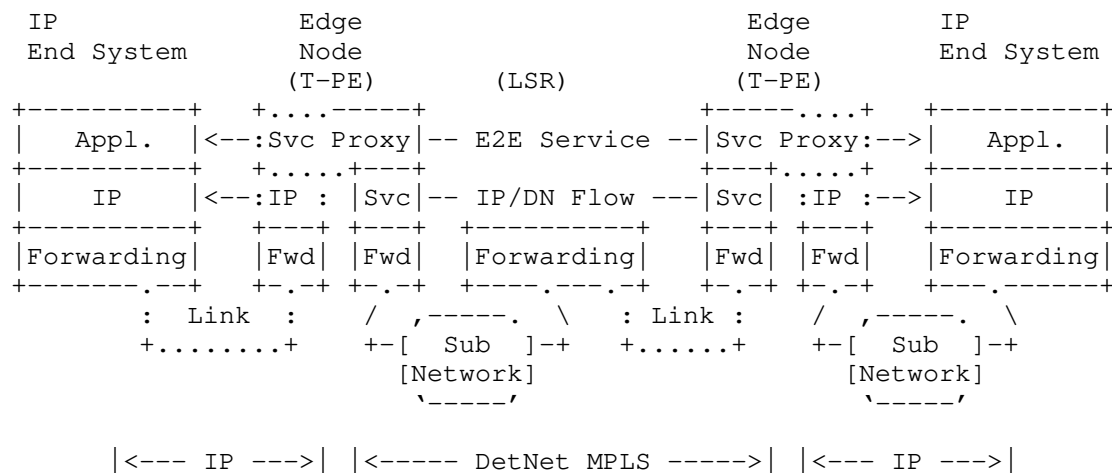


Figure 6: Non-DetNet Aware IP Over DetNet MPLS Network

Figure 6 illustrates non-DetNet enabled End Systems (hosts), connected to DetNet (DN) enabled MPLS network. It differs from Figure 4 in that the hosts and edge IP networks are not DetNet aware. In this case, edge nodes sit at the boundary of the MPLS domain since

it is also a DetNet domain boundary. The edge nodes provide DetNet service proxies for the end applications by initiating and terminating DetNet service for the application's IP flows. See [I-D.ietf-detnet-dp-sol-ip] for more information.

Figure 7 illustrates how it is still possible to provided DetNet service protection for non-DetNet aware end systems. This figures is basically the same as Figure 5, with the exception that CE1 and CE2 are non-DetNet aware end systems and E1 and E3 are edge nodes that replace the relay nodes R1 and R3.

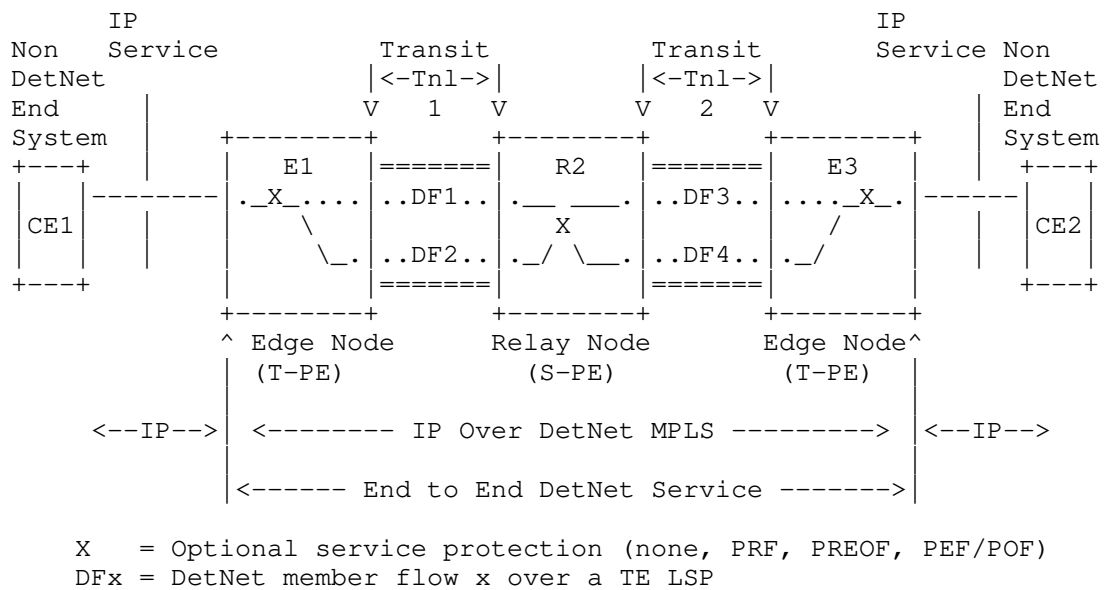


Figure 7: MPLS-Based DetNet (non-MPLS End System)

#### 4.2.2. IEEE 802.1 TSN Over DetNet MPLS Data Plane Scenario

[Author's note: this section to be moved to TSN over mpls sol draft - TBD-TSN-OVER-DETNET]

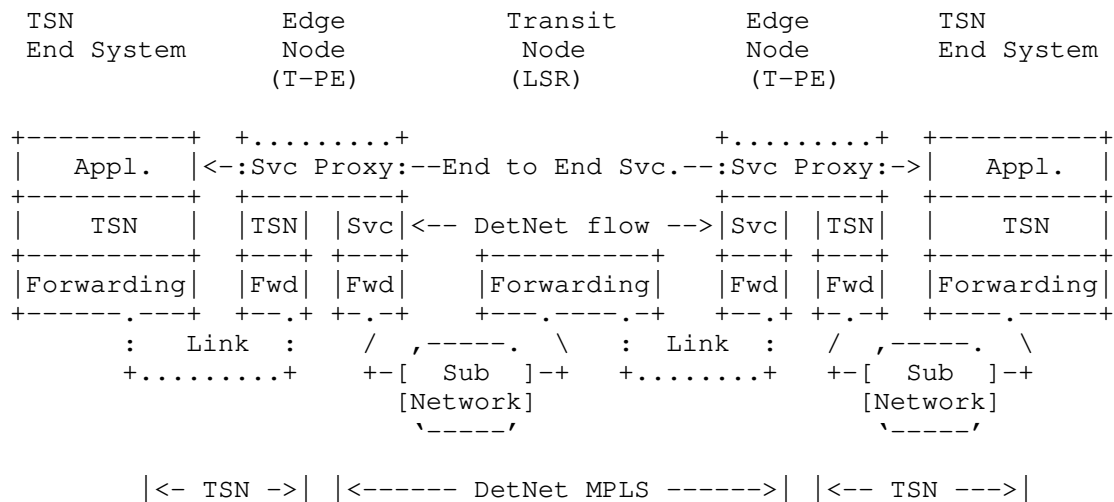


Figure 8: A TSN over DetNet MPLS Enabled Network

Figure 8 shows IEEE 802.1 TSN end stations operating over a TSN aware DetNet service running over an MPLS network. DetNet Edge Nodes sit at the boundary of a DetNet domain. They are responsible for mapping non-DetNet aware L2 traffic to DetNet services. They also support the imposition and disposition of the required DetNet encapsulation. These are functionally similar to pseudowire (PW) Terminating Provider Edge (T-PE) nodes which use MPLS-TE LSPs. In this example they understand and support IEEE 802.1 TSN and are able to map TSN flows into DetNet flows. The specifics of this operation are discussed in [TBD-TSN-OVER-DETNET].

Native TSN flow and DetNet MPLS flow differ not only by the additional MPLS specific encapsulation, but DetNet MPLS flows have on each DetNet node an associated DetNet specific data structure, what defines flow related characteristics and required forwarding functions. In this example, edge nodes provide a service proxy function that "associates" the DetNet flows and native flows at the edge of the DetNet domain. This ensures that the DN Flow is properly served at the Edge node (and inside the domain).

Figure 9 illustrates how DetNet can provide services for IEEE 802.1 TSN end systems, CE1 and CE2, over a DetNet enabled MPLS network. Similar to Figure 6, the edge nodes, E1 and E2, insert and remove required DetNet data plane encapsulation. The 'X' in the edge nodes and relay node, R1, represent a potential DetNet compound flow packet replication and elimination point. This conceptually parallels L2VPN services, and could leverage existing related solutions as discussed below.

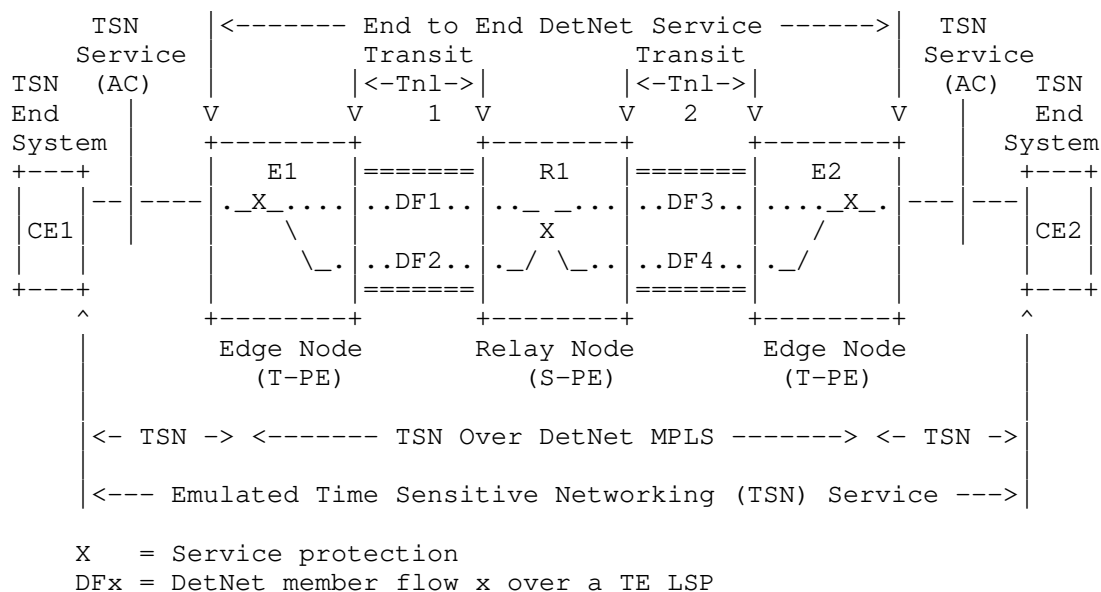


Figure 9: IEEE 802.1TSN Over DetNet

### 4.3. Packet Flow Example with Service Protection

An example DetNet MPLS network fragment and packet flow is illustrated in Figure 10.

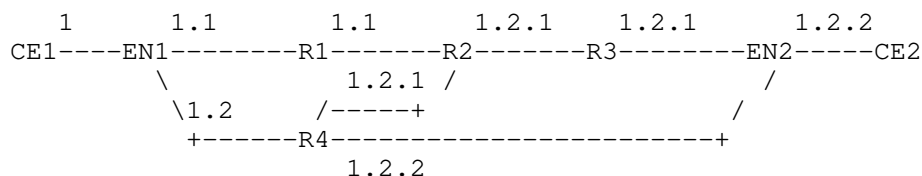


Figure 10: Example Packet Flow in DetNet Enabled MPLS Network

In Figure 10 the numbers are used to identify the instance of a packet. Packet 1 is the original packet, and packets 1.1, and 1.2 are two first generation copies of packet 1. Packet 1.2.1 is a second generation copy of packet 1.2 etc. Note that these numbers never appear in the packet, and are not to be confused with sequence numbers, labels or any other identifier that appears in the packet. They simply indicate the generation number of the original packet so that its passage through the network fragment can be identified to the reader.

Customer Equipment CE1 sends a packet into the DetNet enabled MPLS network. This is packet (1). Edge Node EN1 encapsulates the packet as a DetNet Packet and sends it to Relay node R1 (packet 1.1). EN1 makes a copy of the packet (1.2), encapsulates it and sends this copy to Relay node R4.

Note that along the MPLS path from EN1 to R1 there may be zero or more LSRs which, for clarity, are not shown. The same is true for any other path between two DetNet entities shown in Figure 10.

Relay node R4 has been configured to send one copy of the packet to Relay Node R2 (packet 1.2.1) and one copy to Edge Node EN2 (packet 1.2.2).

R2 receives packet copy 1.2.1 before packet copy 1.1 arrives, and, having been configured to perform packet elimination on this DetNet flow, forwards packet 1.2.1 to Relay Node R3. Packet copy 1.1 is of no further use and so is discarded by R2.

Edge Node EN2 receives packet copy 1.2.2 from R4 before it receives packet copy 1.2.1 from R2 via relay Node R3. EN2 therefore strips any DetNet encapsulation from packet copy 1.2.2 and forwards the packet to CE2. When EN2 receives the later packet copy 1.2.1 this is discarded.

The above is of course illustrative of many network scenarios that can be configured. Between a pair of relay nodes there may be one or more transit nodes that simply forward the DetNet traffic, but these are omitted for clarity.

## 5. DetNet MPLS Data Plane Considerations

This section provides informative considerations related to providing DetNet service to flows which are identified based on their header information. At a high level, the following are provided on a per flow basis:

Eliminating contention loss and jitter reduction:

- Use of allocated resources (queuing, policing, shaping) to ensure that the congestion-related loss and latency/jitter requirements of a DetNet flow are met.

Explicit routes:

- Use of a specific path for a flow. This limits misordering and bounds latency.

#### Service protection:

Which in the case of this document primarily relates to replication and elimination. Changing the explicit path after a failure is detected in order to restore delivery of the required DetNet service characteristics is also possible. Path changes, even in the case of failure recovery, can lead to the out of order delivery of data.

#### Load sharing:

Generally, distributing packets of the same DetNet flow over multiple paths is not recommended. Such load sharing, e.g., via ECMP or UCMP, impacts ordering and possibly jitter.

#### Troubleshooting:

For example, to support identification of misbehaving flows.

#### Recognize flow(s) for analytics:

For example, increase counters.

#### Correlate events with flows:

For example, unexpected loss.

The DetNet data plane also allows for the aggregation of DetNet flows, e.g., via MPLS hierarchical LSPs, to improved scaling. When DetNet flows are aggregated, transit nodes provide service to the aggregate and not on a per-DetNet flow basis. In this case, nodes performing aggregation will ensure that per-flow service requirements are achieved.

### 5.1. End-System Specific Considerations

Data-flows requiring DetNet service are generated and terminated on end-systems. Encapsulation depends on application and its preferences. In a DetNet MPLS domain the DN functions use the d-CWs, S-Labels and F-Labels to provide DetNet services. However, an application may exchange further flow related parameters (e.g., time-stamp), which are not provided by DN functions.

Specifics related to non-MPLS DetNet end station behavior are outside the scope of this document. For example, details on support for DetNet IP data flows can be found in [I-D.ietf-detnet-dp-sol-ip]. This document is also useful for end stations that map IP flows to DetNet flows.

As a general rule, DetNet MPLS domains are capable of forwarding any DetNet MPLS flows and the DetNet domain does not mandate the end-system or edge system encapsulation format. Unless there is a proxy of some form present, end-systems peer with similar end-systems using the same application encapsulation format. For example, as shown in Figure 11, IP applications peer with IP applications and Ethernet L2VPN applications peer with Ethernet L2VPN applications.

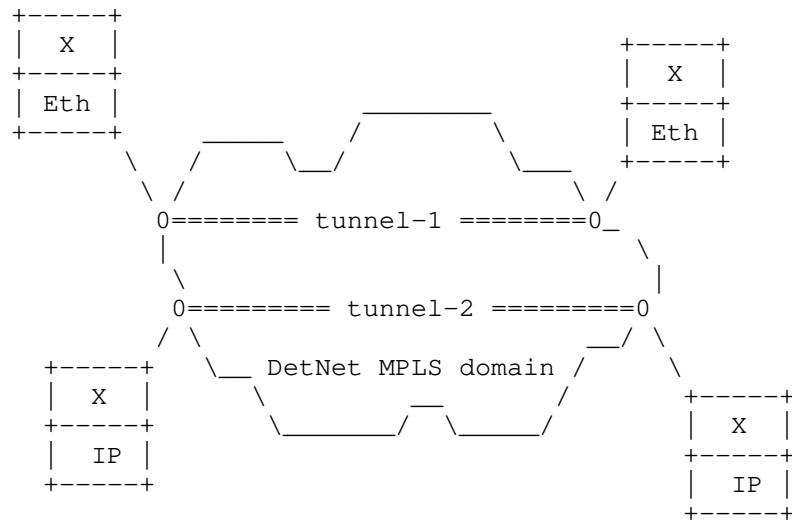


Figure 11: End-Systems and The DetNet MPLS Domain

## 5.2. Sub-Network Considerations

As shown in Figure 2, MPLS nodes are interconnected by different sub-network technologies, which may include point-to-point links. Each of these need to provide appropriate service to DetNet flows. In some cases, e.g., on dedicated point-to-point links or TDM technologies, all that is required is for a DetNet node to appropriately queue its output traffic. In other cases, DetNet nodes will need to map DetNet flows to the flow semantics (i.e., identifiers) and mechanisms used by an underlying sub-network technology. Figure 12 shows several examples of header formats that can be used to carry DetNet MPLS flows over different sub-network technologies. L2 represent a generic layer-2 encapsulation that might be used on a point-to-point link. TSN represents the encapsulation used on an IEEE 802.1 TSN network, as described in Section 8. UDP/IP represents the encapsulation used on a DetNet IP PSN, as described in Section 9.

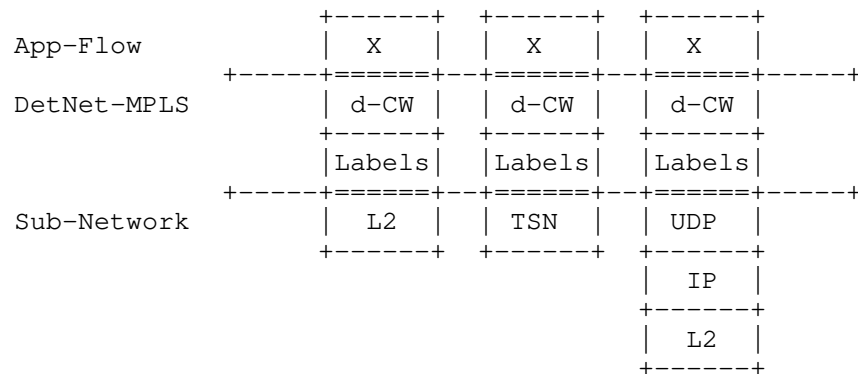


Figure 12: Example DetNet MPLS Sub-Network Formats

## 6. MPLS-Based DetNet Data Plane Solution

### 6.1. DetNet Over MPLS Encapsulation Components

To carry DetNet over MPLS the following is required:

1. A method of identifying the MPLS payload type.
2. A method of identifying the DetNet flow group to the processing element.
3. A method of distinguishing DetNet OAM packets from DetNet data packets.
4. A method of carrying the DetNet sequence number.
5. A suitable LSP to deliver the packet to the egress PE.
6. A method of carrying queuing and forwarding indication.

In this design an MPLS service label (the S-Label), similar to a pseudowire (PW) label [RFC3985], is used to identify both the DetNet flow identity and the payload MPLS payload type satisfying (1) and (2) in the list above. OAM traffic discrimination happens through the use of the Associated Channel method described in [RFC4385]. The DetNet sequence number is carried in the DetNet Control word which carries the Data/OAM discriminator. To simplify implementation and to maximize interoperability two sequence number sizes are supported: a 16 bit sequence number and a 28 bit sequence number. The 16 bit sequence number is needed to support some types of legacy clients. The 28 bit sequence number is used in situations where it is



necessary ensure that in high speed networks the sequence number space does not wrap whilst packets are in flight.

The LSP used to forward the DetNet packet may be of any type (MPLS-LDP, MPLS-TE, MPLS-TP [RFC5921], or MPLS-SR [I-D.ietf-spring-segment-routing-mpls]). The LSP (F-Label) label and/or the S-Label may be used to indicate the queue processing as well as the forwarding parameters. Note that the possible use of Penultimate Hop Popping (PHP) means that the only label in a received label stack may be the S-Label.

## 6.2. MPLS Data Plane Encapsulation

Figure 13 illustrates a DetNet data plane MPLS encapsulation. The MPLS-based encapsulation of the DetNet flows is a good fit for the scenarios described in Section 4.2.1 and Section 4.2.2. Furthermore, end to end DetNet service i.e., native DetNet deployment (see Section 4.2) is also possible if DetNet end systems are capable of initiating and termination MPLS encapsulated packets.

The MPLS-based DetNet data plane encapsulation consists of:

- o DetNet control word (d-CW) containing sequencing information for packet replication and duplicate elimination purposes, and the OAM indicator.
- o DetNet service Label (S-Label) that identifies a DetNet flow at the receiving DetNet service sub-layer processing node.
- o Zero or more Detnet MPLS Forwarding label(s) (F-Label) used to direct the packet along the label switched path (LSP) to the next service sub-layer processing node along the path. When Penultimate Hop Popping is in use there may be no label F-Label in the protocol stack on the final hop.
- o The necessary data-link encapsulation is then applied prior to transmission over the physical media.

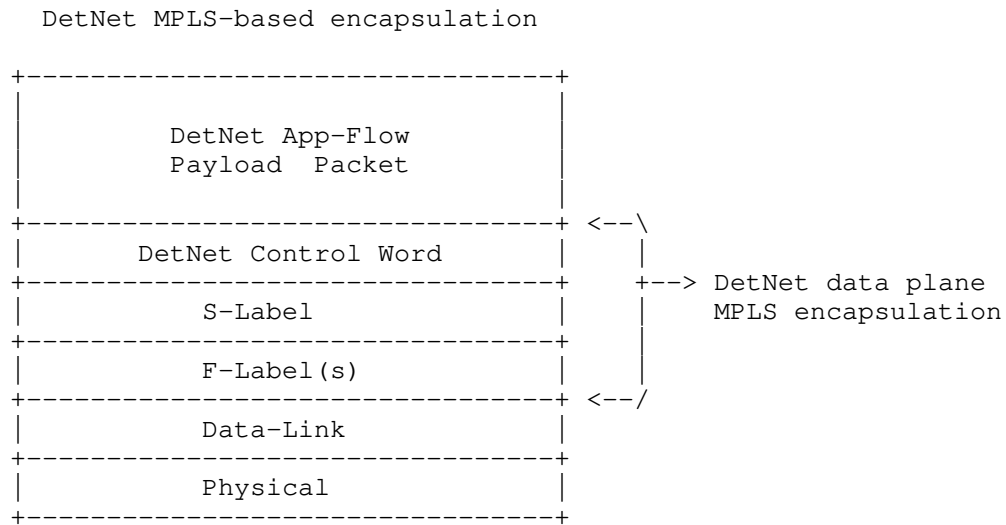


Figure 13: Encapsulation of a DetNet App-Flow in an MPLS(-TP) PSN

#### 6.2.1. DetNet Control Word and the DetNet Sequence Number

A DetNet control word (d-CW) conforms to the Generic PW MPLS Control Word (PWMCW) defined in [RFC4385]. The d-CW formatted as shown in Figure 14 MUST be present in all DetNet packets containing app-flow data.

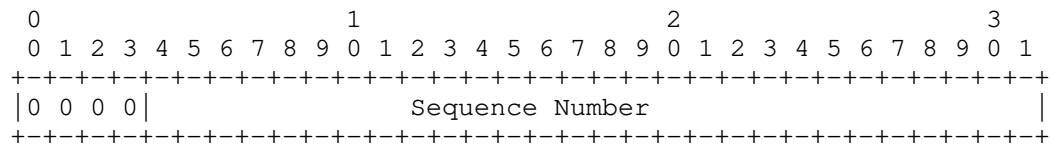


Figure 14: DetNet Control Word

(bits 0 to 3)

Per [RFC4385], MUST be set to zero (0).

Sequence Number (bits 4 to 31)

An unsigned value implementing the DetNet sequence number.

A separate sequence number space MUST be maintained by the the node that adds the d-CW for each DetNet app-flow. The following sequence number field lengths MUST be supported:

0 bits

16 bits

28 bits

The sequence number length MUST be provisioned (configured) on a per app-flow basis via configuration, e.g., the controller plane described in Section 7.

A 0 bit sequence number field length indicates that there is no DetNet sequence number used for the flow. When the length is zero, the sequence number field MUST be set to zero (0) on all packets sent for the flow.

When the sequence number field length is 16 or 28 bits for a flow, the sequence number MUST be incremented by one for each new app-flow packet sent. When the field length is 16 bits, d-CW bits 4 to 15 MUST be set to zero (0). This values carried in this field can wrap and it is important to note that zero (0) is a valid field value. For example, were the sequence number size is 16 bits, the sequence will contain: 65535, 0, 1. In this case, zero (0) is an ordinary sequence number. This differs from [RFC4448] where a sequence number of zero (0) does not indicate that no sequence number field value is in use.

The sequence number is optionally used during receive processing as described below in Section 6.2.2.1 and Section 6.2.2.2.

#### 6.2.2. S-Labels

App-flow identification at a DetNet service sub-layer is realized by an S-Label. Each app-flow MUST be sent by the node that adds a d-CW with a single specific S-Label value. MPLS-aware DetNet end systems and edge nodes, which are by definition MPLS ingress and egress nodes, MUST add and remove the d-CW and S-Label. Relay nodes MAY swap S-Label values when processing an app-flow.

The S-Label value MUST be provisioned per app-flow via configuration, e.g., via the controller plane described in Section 7. Note that S-Labels provide app-flow identification at the downstream DetNet service sub-layer receiver, not the sender. As such, S-Labels MUST be allocated by the entity that controls the service sub-layer

receiving node's label space, and MAY be allocated from the platform label space [RFC3031].

The S-Label will normally be at the bottom of the label stack, immediately preceding the d-CW. To support service sub-layer level OAM, an OAM Associated Channel Header (ACH) [RFC4385] together with a Generic Associated Channel Label (GAL) [RFC5586] MAY be used in place of a d-CW.

Similarly, an Entropy Label Indicator/Entropy Label (ELI/EL) [RFC6790] MAY be carried below the S-Label in the label stack in networks where DetNet flows would otherwise received ECMP treatment. When ELs are used, the same EL value SHOULD be used for all of the packets sent using a specific S-Label to force the flow to follow the same path. However, as previously stated in Section 5, the use of ECMP for DetNet flows is NOT RECOMMENDED. ECMP MAY be used for non-DetNet flows within a DetNet domain.

When receiving a DetNet MPLS flow, an implementation MUST identify the app-flow associated with the incoming packet based on the S-Label. When a node is using platform labels for S-Labels, no additional information is needed as the S-label uniquely identifies the app-flow. In the case where platform labels are not used, one or more F-Labels proceeding the S-Label MUST be used together with the S-Label to uniquely identify the incoming app-flows. When PHP is used, the incoming interface MAY also be used together with any present F-Label(s) and the S-Label to uniquely identify an incoming app-flows. Note that choice to use platform label space for S-Label or S-Label plus one or more F-Labels to identify app flows is a local implementation choice, with one caveat. When one or more F-labels, or incoming interface, is needed together with an S-Label to uniquely identify, the controller plane MUST ensure that incoming DetNet MPLS packets arrive with the needed information (F-label(s) and/or incoming interface); the details of such are outside the scope of this document.

While NOT REQUIRED, the use of platform labels for S-Labels matches other pseudowire encapsulations. This implementation choice also impacts PEF and POF processing as described in the next section.

#### 6.2.2.1. Packet Elimination Function Processing

Implementations MAY support the Packet Elimination Function (PEF) for received DetNet MPLS flows. When supported, use of the PEF for a particular app-flow MUST be provisioned via configuration, e.g., via the controller plane described in Section 7.

After an app-flow is identified for a received DetNet MPLS packet, as described above, an implementation MUST check if PEF is configured for that app-flow. When configured the implementation MUST track the sequence number contained in received d-CWs and MUST ensure that duplicate (replicated) instances of a particular sequence number are discarded. The specific mechanisms used for an implementation to identify which received packets are duplicates and which are new is an implementation choice. Note that per Section 6.2.1 the sequence number field length may be 16 or 28 bits, and the field value can wrap.

Note that an implementation MAY wish to constrain the maximum number sequence numbers that are tracked, on platform-wide or per flow basis. Some implementations MAY support the provisioning of the maximum number sequence numbers that are tracked number on either a platform-wide or per flow basis.

#### 6.2.2.2. Packet Ordering Function Processing

A function that is related to PEF is the Packet Ordering Function (POF). Implementations MAY support POF. When supported, use of the POF for a particular app-flow MUST be provisioned via configuration, e.g., via the controller plane described by Section 7. Implementations MAY require that PEF and POF be used in combination. There is no requirement related to the order of execution of the Packet Elimination and Ordering Functions in an implementation.

After an app-flow is identified for a received DetNet MPLS packet, as described above, an implementation MUST check if POF is configured for that app-flow. When configured the implementation MUST track the sequence number contained in received d-CWs and MUST ensure that packets are processed in the order indicated in the received d-CW sequence number field, which may not be in the order the packets are received. As defined in Section 6.2.1 the sequence number field length may be 16 or 28 bits, is incremented by one (1) for each new app-flow packet sent, and the field value can wrap. The specific mechanisms used for an implementation to identify the order of received packets is an implementation choice.

Note that an implementation MAY wish to constrain the maximum number of out of order packets that can be processed, on platform-wide or per flow basis. Some implementations MAY support the provisioning of this number on either a platform-wide or per flow basis. The number of out of order packets that can be processed also impacts the latency of a flow.

### 6.2.3. F-Labels

F-Labels are support the DetNet forwarding sub-layer. F-Labels are used to provide LSP-based connectivity between DetNet service sub-layer processing nodes.

#### 6.2.3.1. Service Sub-Layer and Packet Replication Function Processing

DetNet MPLS end systems, edge nodes and relay nodes may operate at the DetNet service sub-layer with understand of app-flows and their requirements. As mentioned above, when operating at this layer such nodes can push, pop or swap (pop then push) S-Labels. In all cases, the F-Labels used for the app-flow are always replaced and this section applies.

When sending a DetNet flow, Zero or more F-Labels MAY be added on top of an S-Label by the node pushing an S-Label. The F-Labels to be pushed when sending a particular app-flow MUST be provisioned per app-flow via configuration, e.g., via the controller plane discussed in Section 7. To allow for the omission of F-Labels, an implementation SHOULD also allow an outgoing interface to be provisioned.

The Packet Replication Function (PRF) function MAY be supported by an implementation for outgoing DetNet flows. When supported, the same app-flow data will be sent over multiple outgoing forwarding sub-layer LSPs. To support PRF an implementation MUST support the setting of different sets of F-Labels. Hereto, to allow for the omission of F-Labels, an implementation SHOULD also allow multiple outgoing interfaces to be provisioned. PRF MUST NOT be used with app-flows configured with a d-CW sequence number field length of 0 bits.

When a single set of F-Labels is provisioned for a particular outgoing app-flow, that set of F-labels MUST be pushed after the S-Label is pushed. The outgoing packet is then forwarded as described below in Section 6.2.3.2. When a single outgoing interface is provisioned, the outgoing packet is then forwarded as described below in Section 6.2.3.2.

When multiple sets of F-Labels or interfaces are provisioned for a particular outgoing app-flow, a copy of the outgoing packet, including the pushed S-Label, MUST be made per F-label set and outgoing interface. Each set of provisioned F-Labels are then pushed onto a copy of the packet. Each copy is then forwarded as described below in Section 6.2.3.2.

As described in the previous section, when receiving a DetNet MPLS flow, an implementation identifies the app-flow associated with the incoming packet based on the S-Label. When a node is using platform labels for S-Labels, any F-Labels can be popped and the S-label uniquely identifies the app-flow. In the case where platform labels are not used, F-Label(s) MUST also be provisioned for incoming app-flows. When PHP is used, incoming interface MUST be provisioned. The provisioned information MUST then be used to identify incoming app-flows based on the combination of S-Label and F-Label(s) or incoming interface.

#### 6.2.3.2. Common F-Label Processing

All DetNet aware MPLS nodes process F-Labels as needed to meet the service requirements of the DetNet flow or flows carried in the LSPs represented by the F-Labels. This includes normal push, pop and swap operations. Such processing is essentially the same type of processing enabled for TE LSPs, although the specific service parameters, or traffic specification, can differ. When the DetNet service parameters of the app-flow or flows carried in an LSP represented by an F-Label can be met by an existing TE mechanism, the forwarding sub-layer processing node MAY be a DetNet unaware, i.e., standard, MPLS LSR. Such TE LSPs may provide LSP forwarding service as defined in, but not limited to, [RFC3209], [RFC3270], [RFC3272], [RFC3473], [RFC4875], [RFC5440], and [RFC6006].

More specifically, as mentioned above, the DetNet forwarding sub-layer provides explicit routes and allocated resources, and F-Labels are used to map to each. Explicit routes are supported based on the topmost (outermost) F-Label that is pushed or swapped and the LSP that corresponds to this label. This topmost (outgoing) label MUST be associated with a provisioned outgoing interface and, for non-point-to-point outgoing interfaces, a next hop LSR. Note that this information MUST be provisioned via configuration or the controller plane. In the previously mentioned special case where there is no added F-labels and the outgoing interface is not a point-to-point interface, the outgoing interface MUST also be associated with a next hop LSR.

Resources may be allocated in a hierarchical fashion per LSP that is represented by each F-Label. Each LSP MAY be provisioned with a service parameters that dictates the specific traffic treatment to be received by the traffic carried over that LSP. Implementations of this document MUST ensure that traffic carried over each LSP represented by an F-Label receives the traffic treatment provisioned for that LSP. Typical mechanisms used to provide different treatment to different flows includes the allocation of system resources (such as queues and buffers) and provisioning or related parameters (such

as shaping, and policing). Support can also be provided via an underlying network technology such IEEE802.1 TSN Section 8. Other than in the TSN case, the specific mechanisms used by a DetNet node to ensure DetNet service delivery requirements are met for supported DetNet flows is outside the scope of this document.

Packets that are marked in a way that does not correspond to allocated resources, e.g., lack a provisioned F-Label, can disrupt the QoS offered to properly reserved DetNet flows by using resources allocated to the reserved flows. Therefore, the network nodes of a DetNet network:

- o MUST defend the DetNet QoS by discarding or remarking (to an allocated DetNet flow or non-competing non-DetNet flow) packets received that are not the subject of a completed resource allocation.
- o MUST NOT use a DetNet allocated resource, e.g. a queue or shaper reserved for DetNet flows, for any packet that does match the corresponding DetNet flow.
- o MUST ensure a QoS flow does not exceed its allocated resources or provisioned service level,
- o MUST ensure a CoS flow or service class does not impact the service delivered to other flows. This requirement is similar to requirement for MPLS LSRs to that CoS LSPs do not impact the resources allocated to TE LSPs, e.g., via [RFC3473].

Subsequent sections provide additional considerations related to CoS (Section 6.6.1), QoS (Section 6.6.2) and aggregation (Section 6.6.3).

### 6.3. OAM Indication

OAM follows the procedures set out in [RFC5085] with the restriction that only Virtual Circuit Connectivity Verification (VCCV) type 1 is supported.

As shown in Figure 3 of [RFC5085] when the first nibble of the d-CW is 0x0 the payload following the d-CW is normal user data. However, when the first nibble of the d-CW is 0x1, the payload that follows the d-DW is an OAM payload with the OAM type indicated by the value in the d-CW Channel Type field.

The reader is referred to [RFC5085] for a more detailed description of the Associated Channel mechanism, and to the DetNet work on OAM for more information DetNet OAM.



#### 6.4. Flow Aggregation

[Author's note: need to revisit this section and ensure that we cover (and fully specify) desired types of aggregation.]

1. Aggregate at the LSP (Forwarding)
2. Aggregating DetNet flows as a new DetNet flow
3. Simple Aggregation at the DetNet layer

The resource control and management aspects of aggregation (including the queuing/shaping/ policing implications) will be covered in other documents.

The ability to aggregate individual flows, and their associated resource control, into a larger aggregate is an important technique for improving scaling of control in the data, management and control planes. The DetNet data plane allows for the aggregation of DetNet flows, to improved scaling. There are three methods of introducing flow aggregation:

[Editor's note:]

The following review comments were received when this section was committed to github.

General comment: We should points to the major issue of aggregation, namely the Seq.Num related problem. The aggregated flows have their own Seq.Num and those are independent. We should consider to group the aggregation techniques as per their impact on what DetNet functions they allow on a DetNet flow. (E.g., aggregation without new Aggregate.Seq.Num would prohibit usage of FR, EF and in-order-delivery function on the aggregate flow).

SR based aggregation can be treated as a form of H-LSP aggregation. Should we differentiate them? What are the differences?

What are the issues when aggregating of different payload types? Should we add an editor note on this?

Simple-aggregation-at-the-detnet-layer: is this not the same as H-LSP? The A-label can be treated just as an additional F-Label.

[Editor's note: End of review comment.]

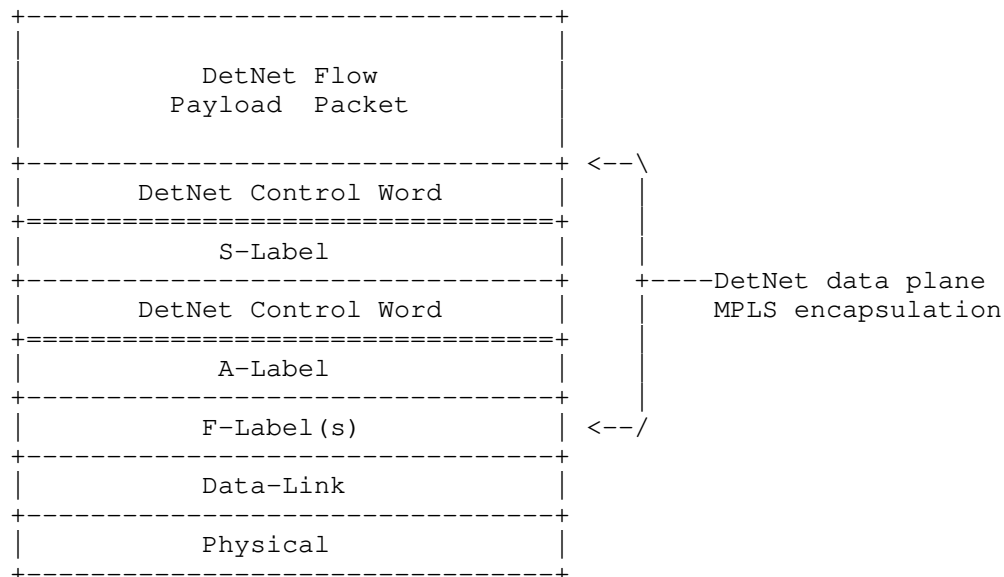
#### 6.4.1. Aggregation at the LSP

DetNet flows forwarded via MPLS can leverage MPLS-TE's existing support for hierarchical LSPs (H-LSPs), see [RFC4206]. H-LSPs are typically used to aggregate control and resources, they may also be used to provide OAM or protection for the aggregated LSPs. Arbitrary levels of aggregation naturally falls out of the definition for hierarchy and the MPLS label stack [RFC3032]. DetNet nodes which support aggregation (LSP hierarchy) map one or more LSPs (labels) into and from an H-LSP. Both carried LSPs and H-LSPs may or may not use the TC field, i.e., L-LSPs or E-LSPs. Such nodes will need to ensure that traffic from aggregated LSPs are placed (shaped/policed/enqueued) onto the H-LSPs in a fashion that ensures the required DetNet service is preserved.

Additional details of the traffic control capabilities needed at a DetNet-aware node may be covered in the new service descriptions mentioned above or in separate future documents. Management and control plane mechanisms will also need to ensure that the service required on the aggregate flow (H-LSP or DSCP) are provided, which may include the discarding or remarking mentioned in the previous sections.

#### 6.4.2. Aggregating DetNet Flows as a new DetNet flow

An aggregate can be built by layering DetNet flows as shown below:

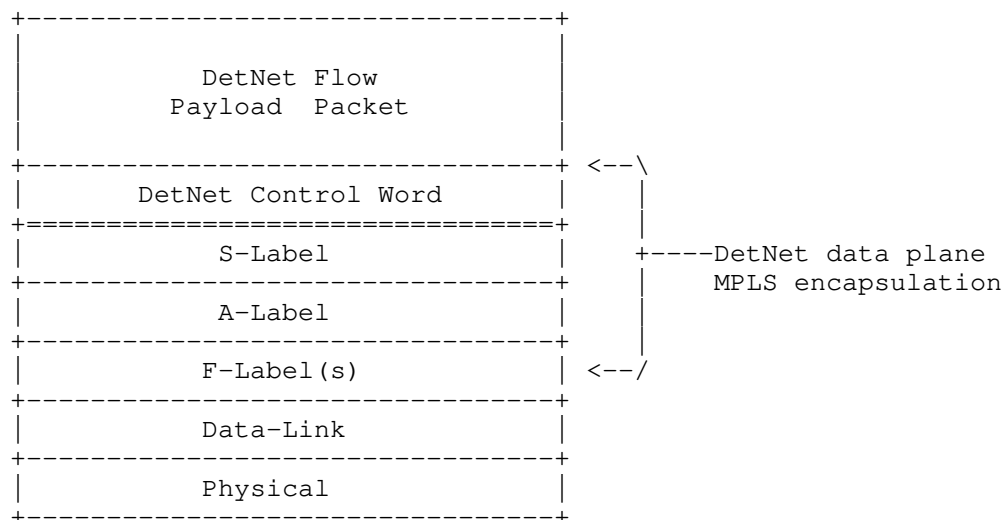


Both the Aggregation (A) label and the S-Label have their MPLS S bit set indicating bottom of stack, and the d-CW allows the PREOF to work.

It is a property of the A-label that what follows is d-CW followed by an S-Label. A relay node processing the A-label would not know the underlying payload type. This would only be known to a node that was a peer of the node imposing the S-Label. However there is no real need for it to know the payload type during aggregation processing.

#### 6.4.3. Simple Aggregation at the DetNet Layer

Another approach would be not to include a d-CW for the aggregated flow. This would be functionally similar to aggregation at the forwarding sub-layer using H-LSPs, but would confine knowledge of the aggregation to the DetNet layer. Such an approach shares the disadvantage that PREOF operations would not be possible. OAM operation in this mode is for further study.



#### 6.5. Service Sub-Layer Considerations

The edge and relay node internal procedures related to PREOF are implementation specific. The order of a packet elimination or replication is out of scope in this specification. However, care should be taken that the replication function does not actually loopback packets as "replicas". Looped back packets include artificial delay when the node that originally initiated the packet receives it again. Also, looped back packets may make the network condition to look healthier than it actually is (in some cases link

failures are not reflected properly because looped back packets make the situation appear better than it actually is).

It is important that the DetNet layer is configured such that a DetNet node never receives its own replicated packets. If it were to receive such packets the replication function would make the loop more destructive of bandwidth than a conventional unicast loop. Ultimately the TTL in the S-Label will cause the packet to die during a transient, but given the sensitivity of applications to packet latency the impact on the DetNet application would be severe.

#### 6.5.1. Edge Node Processing

An edge node is responsible for matching ingress packets to the service they require and encapsulating them accordingly. An edge node may participate in the packet replication and duplication elimination.

The DetNet-aware forwarder selects the egress DetNet member flow segment based on the flow identification. The mapping of ingress DetNet member flow segment to egress DetNet member flow segment may be statically or dynamically configured. Additionally the DetNet-aware forwarder does duplicate frame elimination based on the flow identification and the sequence number combination. The packet replication is also done within the DetNet-aware forwarder. During elimination and the replication process the sequence number of the DetNet member flow MUST be preserved and copied to the egress DetNet member flow.

The internal design of a relay node is out of scope of this document. However the reader's attention is drawn to the need to make any PREOF state available to the packet processor(s) dealing with packets to which the PREOF functions must be applied, and to maintain that state in such a way that it is available to the packet processor operation on the next packet in the DetNet flow (which may be a duplicate, a late packet, or the next packet in sequence).

[Editor's note: I think the rest of this section belongs in a new "802.1 TSN (island Interconnect) over DetNet MPLS" section.]

This may be done in the DetNet layer, or where the native service processing (NSP) [RFC3985] is IEEE 802.1CB [IEEE8021CB] capable, the packet replication and duplicate elimination MAY entirely be done in the NSP, bypassing the DetNet flow encapsulation and logic entirely. This enables operating over unmodified implementations and deployments. The NSP approach works only between edge nodes and cannot make use of relay nodes.

The NSP approach is useful end to end tunnel and for for "island interconnect" scenarios. However, when there is a need to do PREOF in a middle of the network, such plain edge to edge operation is not sufficient.

The extended forwarder MAY copy the sequencing information from the native DetNet packet into the DetNet sequence number field and vice versa. If there is no existing sequencing information available in the native packet or the forwarder chose not to copy it from the native packet, then the extended forwarder MUST maintain a sequence number counter for each DetNet flow (indexed by the DetNet flow identification).

#### 6.5.2. Relay Node Processing

A DetNet Relay node operates in the DetNet forwarding sub-layer . This processing is done within an extended forwarder function. Whether an ingress DetNet member flow receives DetNet specific processing depends on how the forwarding is programmed. Some relay nodes may be DetNet service aware, while others may be unmodified LSRs that only understand how to swicth MPLS-TE LSPs.

It is also possible to treat the relay node as a transit node, see Section 6.6.3. Again, this is entirely up to how the forwarding has been programmed.

#### 6.6. Forwarding Sub-Layer Considerations

##### 6.6.1. Class of Service

Class and quality of service, i.e., CoS and QoS, are terms that are often used interchangeably and confused with each other. In the context of DetNet, CoS is used to refer to mechanisms that provide traffic forwarding treatment based on aggregate group basis and QoS is used to refer to mechanisms that provide traffic forwarding treatment based on a specific DetNet flow basis. Examples of existing network level CoS mechanisms include DiffServ which is enabled by IP header differentiated services code point (DSCP) field [RFC2474] and MPLS label traffic class field [RFC5462], and at Layer-2, by IEEE 802.1p priority code point (PCP).

CoS for DetNet flows carried in PWs and MPLS is provided using the existing MPLS Differentiated Services (DiffServ) architecture [RFC3270]. Both E-LSP and L-LSP MPLS DiffServ modes MAY be used to support DetNet flows. The Traffic Class field (formerly the EXP field) of an MPLS label follows the definition of [RFC5462] and [RFC3270]. The Uniform, Pipe, and Short Pipe DiffServ tunneling and TTL processing models are described in [RFC3270] and [RFC3443] and

MAY be used for MPLS LSPs supporting DetNet flows. MPLS ECN MAY also be used as defined in ECN [RFC5129] and updated by [RFC5462].

#### 6.6.2. Quality of Service

In addition to explicit routes, and packet replication and elimination, described in Section 6 above, DetNet provides zero congestion loss and bounded latency and jitter. As described in [I-D.ietf-detnet-architecture], there are different mechanisms that maybe used separately or in combination to deliver a zero congestion loss service. This includes Quality of Service (QoS) mechanisms at the MPLS layer, that may be combined with the mechanisms defined by the underlying network layer such as 802.1TSN.

Quality of Service (QoS) mechanisms for flow specific traffic treatment typically includes a guarantee/agreement for the service, and allocation of resources to support the service. Example QoS mechanisms include discrete resource allocation, admission control, flow identification and isolation, and sometimes path control, traffic protection, shaping, policing and remarking. Example protocols that support QoS control include Resource ReSerVation Protocol (RSVP) [RFC2205] (RSVP) and RSVP-TE [RFC3209] and [RFC3473]. The existing MPLS mechanisms defined to support CoS [RFC3270] can also be used to reserve resources for specific traffic classes.

A baseline set of QoS capabilities for DetNet flows carried in PWs and MPLS can provided by MPLS with Traffic Engineering (MPLS-TE) [RFC3209] and [RFC3473]. TE LSPs can also support explicit routes (path pinning). Current service definitions for packet TE LSPs can be found in "Specification of the Controlled Load Quality of Service", [RFC2211], "Specification of Guaranteed Quality of Service", [RFC2212], and "Ethernet Traffic Parameters", [RFC6003]. Additional service definitions are expected in future documents to support the full range of DetNet services. In all cases, the existing label-based marking mechanisms defined for TE-LSPs and even E-LSPs are use to support the identification of flows requiring DetNet QoS.

#### 6.6.3. Cross-DetNet Flow Resource Aggregation

[Editor's NOTE: Isn't this section the same as "Aggregation at the LSP". -- Address as part of aggregation section cleanup.]

The ability to aggregate individual flows, and their associated resource control, into a larger aggregate is an important technique for improving scaling of control in the data, management and control planes. This document identifies the traffic identification related aspects of aggregation of DetNet flows. The resource control and

management aspects of aggregation (including the queuing/shaping/policing implications) will be covered in other documents. The data plane implications of aggregation are independent for PW/MPLS and IP encapsulated DetNet flows.

DetNet flows forwarded via MPLS can leverage MPLS-TE's existing support for hierarchical LSPs (H-LSPs), see [RFC4206]. H-LSPs are typically used to aggregate control and resources, they may also be used to provide OAM or protection for the aggregated LSPs. Arbitrary levels of aggregation naturally falls out of the definition for hierarchy and the MPLS label stack [RFC3032]. DetNet nodes which support aggregation (LSP hierarchy) map one or more LSPs (labels) into and from an H-LSP. Both carried LSPs and H-LSPs may or may not use the TC field, i.e., L-LSPs or E-LSPs. Such nodes will need to ensure that traffic from aggregated LSPs are placed (shaped/policed/enqueued) onto the H-LSPs in a fashion that ensures the required DetNet service is preserved.

[NOTE: This needs to be revised:] Additional details of the traffic control capabilities needed at a DetNet-aware node may be covered in the new service descriptions mentioned above or in separate future documents. Management and control plane mechanisms will also need to ensure that the service required on the aggregate flow (H-LSP or DSCP) are provided, which may include the discarding or remarking mentioned in the previous sections.

#### 6.6.4. Layer 2 Addressing and QoS Considerations

[Editor's NOTE: review and simplify this section. Doesn't this belong in the TSN section? Alternatively, describe in generic/non sub-network technology specific terms.]

The Time-Sensitive Networking (TSN) Task Group of the IEEE 802.1 Working Group have defined (and are defining) a number of amendments to IEEE 802.1Q [IEEE8021Q] that provide zero congestion loss and bounded latency in bridged networks. IEEE 802.1CB [IEEE8021CB] defines packet replication and elimination functions that should prove both compatible with and useful to, DetNet networks.

As is the case for DetNet, a Layer 2 network node such as a bridge may need to identify the specific DetNet flow to which a packet belongs in order to provide the TSN/DetNet QoS for that packet. It also will likely need a CoS marking, such as the priority field of an IEEE Std 802.1Q VLAN tag, to give the packet proper service.

Although the flow identification methods described in IEEE 802.1CB [IEEE8021CB] are flexible, and in fact, include IP 5-tuple identification methods, the baseline TSN standards assume that every

Ethernet frame belonging to a TSN stream (i.e. DetNet flow) carries a multicast destination MAC address that is unique to that flow within the bridged network over which it is carried. Furthermore, IEEE 802.1CB [IEEE8021CB] describes three methods by which a packet sequence number can be encoded in an Ethernet frame.

Ensuring that the proper Ethernet VLAN tag priority and destination MAC address are used on a DetNet/TSN packet may require further clarification of the customary L2/L3 transformations carried out by routers and edge label switches. Edge nodes may also have to move sequence number fields among Layer 2, PW, and IP encapsulations.

#### 6.6.5. Time Synchronization

[Editor's Note: A detailed discussion of time synchronization is outside the scope of this document, and the production of a specialist text discussing this topic is encouraged. This section will be updated/removed if such a document is available before publication of this text.]

Time synchronization is important both from the perspective of operating the DetNet network itself and from the perspective of transferring time across the network between client applications. Some clients may be able to use the DetNet as their provider of time and frequency, others may require the DetNet to transfer time between a client clock source and a client clock user.

For example, [RFC8169] describes a method of recording the packet queuing time in an MPLS LSR on a packet by per packet basis and forwarding this information to the egress edge system. This allows compensation for any variable packet queuing delay to be applied at the packet receiver. Other mechanisms for IP/MPLS networks are defined based on IEEE Standard 1588 [IEEE1588], such as ITU-T [G.8275.1] and [G.8275.2].

A more detailed discussion of time synchronization is outside the scope of this document.

### 7. Controller Plane (Management and Control) Considerations

While management plane and control planes are traditionally considered separately, from the Data Plane perspective there is no practical difference based on the origin of flow provisioning information, and the DetNet architecture [I-D.ietf-detnet-architecture] refers to these collectively as the 'Controller Plane'. This document therefore does not distinguish between information provided by distributed control plane protocols, e.g., RSVP-TE [RFC3209] and [RFC3473], or by centralized network



management mechanisms, e.g., RestConf [RFC8040], YANG [RFC7950], and the Path Computation Element Communication Protocol (PCEP) [I-D.ietf-pce-pcep-extension-for-pce-controller] or any combination thereof. Specific considerations and requirements for the DetNet Controller Plane are discussed in Section 7.6.

#### 7.1. S-Label and F-Label Assignment and Distribution

[Editor's note - we may need additional text on resource allocation in this section.]

DetNet S-Labels (see Section 6.2.2 for their definition) are similar to other MPLS service labels that denote the contents of the MPLS packet payload such as a layer 2 pseudowire, an IP packet that is routed in a VPN context with a private address, or an Ethernet virtual private network (EVPN) service.

S-Labels are expected to be allocated in the same manner as any other service labels. S-Labels uniquely identify a particular DetNet flow, and are local to the node on which the label is allocated. In the DetNet service sub-layer the explicit route consists of the set of Relay Nodes that the DetNet flow must traverse. They can be used to identify the DetNet flow that a packet belongs to as it traverses a particular node in a DetNet domain. Because labels are local to each node rather than being a global identifier within a domain, they must be advertised to their upstream DetNet service-aware peer nodes (e.g., a DetNet MPLS End System as shown in Figure 3, or a DetNet Relay or Edge Node as shown in Figure 7) and interpreted in the context of their received F-Label.

As discussed in Section 4, the forwarding sub-layer uses one or more F-Labels to forward DetNet packets between DetNet service-aware nodes along explicitly defined routes at the DetNet forwarding sub-layer, which in the context of this document is the MPLS layer. F-Labels can also provide context for an S-Label. In the DetNet Forwarding (MPLS) sub-layer the explicit route consists of the set of DetNet nodes which are LSRs, links, and possibly link bundle members and queues that the DetNet packets of a flow must traverse between nodes in the DetNet service sub-layer (i.e. between a specific Edge Node and the next hop Relay Node, between specific Relay Nodes, and between a specific Relay node and the egress Edge Node. Resource allocation corresponding to the set of Services supported over the forwarding sub-layer, which may or may not include aggregation, is required at this sub-layer. Explicit routes are used to ensure that packets are routed through the resources that have been reserved for them, and hence provide the DetNet application with the required service. Multiple F-Labels may be pushed after an S-Label and there is no requirement for all F-Labels to be controlled via the same

controller mechanisms. For example in EVPN, some labels are distributed using BGP while others are distributed using LDP or RSVP.

Whether configuring, calculating and instantiating these routes is a single-stage or multi-stage process, or in a centralized or distributed manner, is out of scope of this document.

There are a number of approaches that could be used to provide explicit routes and resource allocation in the MPLS layer:

- o The path could be explicitly set up by a controller which calculates the path and explicitly configures each node along that path with the appropriate forwarding and resource allocation information.
- o The path could be set up using RSVP-TE signaling.
- o The path could be implemented using MPLS-based segment routing when extended to support resource allocation.

See Section 7.6 for further discussion of these alternatives.

Much like other MPLS labels, there are a number of alternatives available for DetNet S-Label and F-Label advertisement to an upstream peer node. These include distributed signaling protocols such as RSVP-TE, centralized label distribution via a controller that manages both the sender and the receiver using NETCONF/YANG, BGP, PCEP, etc., and hybrid combinations of the two. The details of the controller plane solution required for the label distribution and the management of the label number space are out of scope of this document, but as mentioned above, there are particular DetNet considerations and requirements that are discussed in Section 7.6.

## 7.2. Packet Replication, Elimination, and Ordering (PREOF)

The controller plane protocol solution required for managing the PREOF processing is outside the scope of this document. That said, it should be noted that the ability to determine, for a particular flow, optimal packet replication and elimination points in the DetNet domain requires explicit support. There are capabilities that can be used, or extended, for example GMPLS end-to-end recovery [RFC4872] and GMPLS segment recovery [RFC4873].

## 7.3. Contention Loss and Jitter Reduction

As discussed in Section 1, this document does not specify the mechanisms needed to eliminate contention loss or reduce jitter for DetNet flows at the DetNet forwarding sub-layer. The ability to

manage node and link resources to be able to provide these functions will be a necessary part of the DetNet controller plane. It will also be necessary to be able to control the required queuing mechanisms used to provide these functions along a flow's path through the network. See Section 7.6 for further discussion of these requirements.

#### 7.4. Bidirectional Traffic

Some DetNet applications generate bidirectional traffic. Using MPLS definitions [RFC5654] there are associated bidirectional flows, and co-routed bidirectional flows. MPLS defines a point-to-point associated bidirectional LSP as consisting of two unidirectional point-to-point LSPs, one from A to B and the other from B to A, which are regarded as providing a single logical bidirectional forwarding path. This would be analogous of standard IP routing, or PWs running over two reciprocal unidirectional LSPs. MPLS defines a point-to-point co-routed bidirectional LSP as an associated bidirectional LSP which satisfies the additional constraint that its two unidirectional component LSPs follow the same path (in terms of both nodes and links) in both directions. An important property of co-routed bidirectional LSPs is that their unidirectional component LSPs share fate. In both types of bidirectional LSPs, resource reservations may differ in each direction. The concepts of associated bidirectional flows and co-routed bidirectional flows can be applied to DetNet flows.

While the MPLS data plane must support bidirectional DetNet flows, there are no special bidirectional features with respect to the data plane other than the need for the two directions of a co-routed bidirectional flow to take the same path. Fate sharing and associated vs co-routed bidirectional flows can be managed at the control level. Note that there is no stated requirement for bidirectional DetNet flows to be supported using the same MPLS Labels in each direction.

DetNet's use of PREOF may increase the complexity of using co-routing bidirectional flows, since if PREOF is used, then the replication points in one direction would have to match the elimination points in the other direction, and vice versa, and the optimal points for these functions in one direction may not match the optimal points in the other.

Control and management mechanisms will need to support bidirectional flows, but the specification of such mechanisms are out of scope of this document. Related control plan mechanisms have been defined in [RFC3473], [RFC6387] and [RFC7551].

This is further discussed in Section 7.6.

#### 7.5. Flow Aggregation Control

Section 6.4 discusses the use of flow aggregation in DetNet. It includes flow aggregation accomplished through the use of hierarchical LSPs, aggregating multiple DetNet flows into a single new DetNet flow, and simple aggregation at the DetNet layer. It will be the responsibility of the DetNet controller plane to be able to properly provision the use of these mechanisms. These requirements are included in the next section.

#### 7.6. DetNet Controller (Control and Management) Plane Requirements

While the definition of controller plane for DetNet is out of the scope of this document, there are particular considerations and requirements for such that result from the unique characteristics of the DetNet architecture [I-D.ietf-detnet-architecture] and data plane as defined herein.

The primary requirements of the DetNet controller plane are that it must be able to:

- o Instantiate DetNet flows in a DetNet domain (which may include some or all of explicit path and PREOF replication and elimination node determination, link bandwidth reservations, 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.
- o Manage DetNet S-Label and F-Label allocation and distribution, when the DetNet MPLS encapsulation is in use
- o The ability to support DetNet flow aggregation
- o 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)
- o Scale to handle the number of DetNet flows expected in a domain (which may require per-flow signaling or provisioning)
- o Provision flow identification information at each of the nodes along the path, and it may differ depending on the location in the network and the DetNet functionality (e.g. transit node vs. relay node).

These requirements, as stated earlier, could be satisfied using distributed control protocol signaling (such as RSVP-TE), centralized network management provisioning mechanisms (such as BGP, PCEP, YANG [I-D.ietf-detnet-flow-information-model], etc.) or hybrid combinations of the two, and could also make use of MPLS-based segment routing.

In the abstract, the results of either distributed signaling or centralized provisioning are equivalent from a DetNet data plane perspective – flows are instantiated, explicit routes are determined, resources are reserved, and packets are forwarded through the domain using the MPLS data plane.

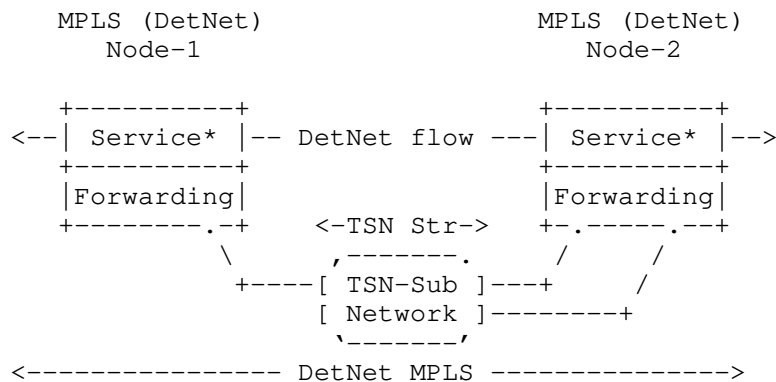
However, from a practical and implementation standpoint, they are not equivalent at all. Some approaches are more scalable than others in terms of signaling load on the network. Some can take advantage of global tracking of resources in the DetNet domain for better overall network resource optimization. Some are more resilient than others if link, node, or management equipment failures occur. While a detailed analysis of the control plane alternatives is out of the scope of this document, the requirements from this document can be used as the basis of a later analysis of the alternatives.

#### 8. DetNet MPLS Operation Over IEEE 802.1 TSN Sub-Networks

[Editor's note: this is a place holder section. A standalone section on MPLS over IEEE 802.1 TSN. Includes RFC2119 Language.]

This section covers how DetNet MPLS flows operate over an IEEE 802.1 TSN sub-network. Figure 15 illustrates such a scenario, where two MPLS (DetNet) nodes are interconnected by a TSN sub-network. Node-1 is single homed and Node-2 is dual-homed. MPLS nodes can be (1) DetNet MPLS End System, (2) DetNet MPLS Edge or Relay node or (3) MPLS Transit node.

Note: in case of MPLS Transit node there is no DetNet Service sub-layer processing.



Note: \* no service sub-layer required for transit nodes

Figure 15: DetNet Enabled MPLS Network Over a TSN Sub-Network

The Time-Sensitive Networking (TSN) Task Group of the IEEE 802.1 Working Group have defined (and are defining) a number of amendments to IEEE 802.1Q [IEEE8021Q] that provide zero congestion loss and bounded latency in bridged networks. Furthermore IEEE 802.1CB [IEEE8021CB] defines frame replication and elimination functions for reliability that should prove both compatible with and useful to, DetNet networks. All these functions have to identify flows those require TSN treatment.

As is the case for DetNet, a Layer 2 network node such as a bridge may need to identify the specific DetNet flow to which a packet belongs in order to provide the TSN/DetNet QoS for that packet. It also may need a CoS marking, such as the priority field of an IEEE Std 802.1Q VLAN tag, to give the packet proper service.

The challenge for MPLS DeNet flows is that the protocol interworking function defined in IEEE 802.1CB [IEEE8021CB] works only for IP flows. The aim of the protocol interworking function is to convert an ingress flow to use a specific multicast destination MAC address and VLAN, for example to direct the packets through a specific path inside the bridged network. A similar interworking pair at the other end of the TSN sub-network would restore the packet to its original destination MAC address and VLAN.

As protocol interworking function defined in [IEEE8021CB] does not work for MPLS labeled flows, the DetNet MPLS nodes MUST ensure proper TSN sub-network specific Ethernet encapsulation of the DetNet MPLS packets. For a given TSN Stream (i.e., DetNet flow) an MPLS (DetNet) node MUST behave as a TSN-aware Talker or a Listener inside the TSN sub-network.

### 8.1. Mapping of TSN Stream ID and Sequence Number

TSN capable MPLS (DetNet) nodes are TSN-aware Talker/Listener as shown in Figure 16. MPLS (DetNet) node MUST provide the TSN sub-network specific Ethernet encapsulation over the link(s) towards the sub-network. An TSN-aware MPLS (DetNet) node MUST support the following TSN components:

1. For recognizing flows:
  - \* Stream Identification (MPLS-flow-aware)
2. For FRER used inside the TSN domain, additionally:
  - \* Sequencing function (MPLS-flow-aware)
  - \* Sequence encode/decode function
3. For FRER when the node is a TSN replication or elimination point, additionally:
  - \* Stream splitting function
  - \* Individual recovery function

[Editor's note: Should we added here requirements regarding IEEE 802.1Q C-VLAN component?]

The Stream Identification and The Sequencing functions are slightly modified for frames passed down the protocol stack from the upper layers.

Stream Identification MUST pair MPLS flows and TSN Streams and encode that in data plane formats as well. The packet's stream\_handle subparameter (see IEEE 802.1CB [IEEE8021CB]) inside the Talker/Listener is defined based on the Flow-ID used in the upper DetNet MPLS layer. Stream Identification function MUST encode Ethernet header fields namely (1) the destination MAC-address, (2) the VLAN-ID and (3) priority parameters with TSN sub-network specific values. Encoding is provided for the frame passed down the stack from the upper layers.

The sequence generation function resides in the Sequencing function. It generates a sequence\_number subparameter for each packet of a Stream passed down to the lower layers. Sequencing function MUST copy sequence information from the MPLS d-CW of the packet to the sequence\_number subparameter for the frame passed down the stack from the upper layers.

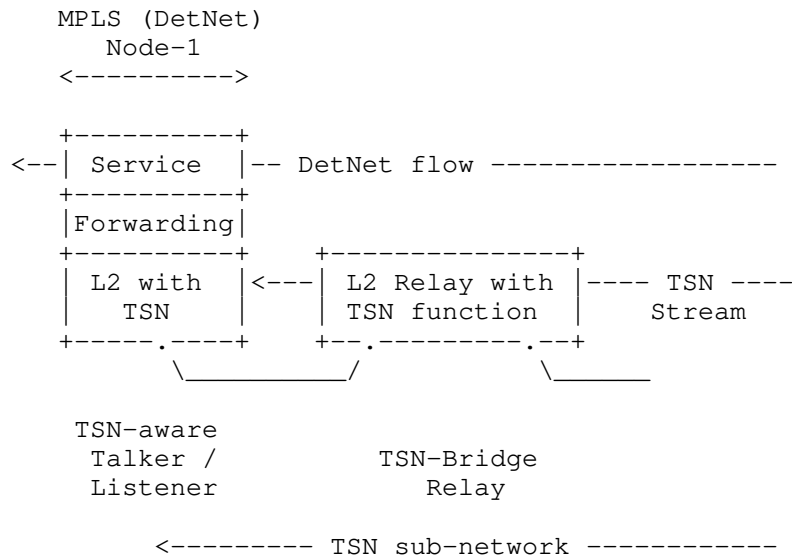


Figure 16: MPLS (DetNet) Node with TSN Functions

The Sequence encode/decode function MUST support the Redundancy tag (R-TAG) format as per Clause 7.8 of IEEE 802.1CB [IEEE8021CB].

## 8.2. TSN Usage of FRER

TSN Streams supporting DetNet flows may use Frame Replication and Elimination for Redundancy (FRER) [802.1CB] based on the loss service requirements of the TSN Stream, which is derived from the DetNet service requirements of the DetNet mapped flow. The specific operation of FRER is not modified by the use of DetNet and follows IEEE 802.1CB [IEEE8021CB].

FRER function and the provided service recovery is available only within the TSN sub-network however as the Stream-ID and the TSN sequence number are paired with the MPLS flow parameters they can be combined with PREOF functions.

## 8.3. Management and Control Implications

[Editor's note: This section is TBD Covers Creation, mapping, removal of TSN Stream IDs, related parameters and,when needed, configuration of FRER. Supported by management/control plane.]



## 9. DetNet MPLS Operation over DetNet IP PSNs

This section specifies the DetNet encapsulation over an IP network. The approach is modeled on the operation of MPLS and PseudoWires (PW) over an IP Packet Switched Network (PSN) [RFC3985][RFC4385][RFC7510]. It maps the MPLS data plane encapsulation described in Section 6.2 to the DetNet IP data plane define in [I-D.ietf-detnet-dp-sol-ip].

To carry DetNet with full functionality at the DetNet layer over an IP network, the following components are required (these are a subset of the requirements for MPLS encapsulation listed in Section 6.1):

1. A method of identifying the DetNet flow group to the processing element.
2. A method of carrying the DetNet sequence number.
3. A method of distinguishing DetNet OAM packets from DetNet data packets.
4. A method of carrying queuing and forwarding indication.

These requirements are satisfied by the DetNet over MPLS Encapsulation described in Section 6.2.

This document builds on the the specification of MPLS over UDP and IP defined in [RFC7510]. It replaces the the F-Label(s) used in Section 6.2 with UDP and IP headers. The UDP and IP header information is used to identify DetNet flows, including member flows, per [I-D.ietf-detnet-dp-sol-ip]. The resulting encapsulation is shown in Figure 17.

Note that this encapsulation works equally well with IPv4 and IPv6.

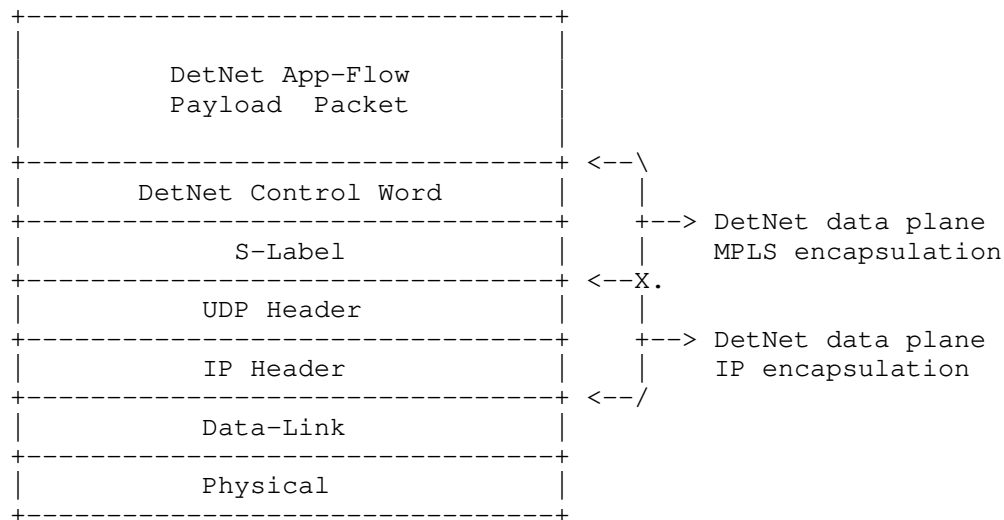


Figure 17: IP Encapsulation of DetNet MPLS

d-CW and S-Labels are used as defined in Section 6.2 and are not modified by this section.

To support outgoing DetNet MPLS over IP, an implementation MUST support the provisioning of IP/UDP header information in place of sets of F-Labels. Note that multiple sets of F-Labels can be provisioned to support PRF on transmitted DetNet flows and therefore, when PRF is supported, multiple IP/UDP headers MAY be provisioned. When multiple IP/UDP headers are provisioned for a particular outgoing app-flow, a copy of the outgoing packet, including the pushed S-Label, MUST be made for each. The headers for each outgoing packet MUST be based on the configuration information and as defined in [RFC7510], with one exception. The one exceptions is that the UDP Source Port value MUST be set to uniquely identify the DetNet (forwarding sub-layer) flow. The packet MUST then be handed as a DetNet IP packet, per [I-D.ietf-detnet-dp-sol-ip].

To support receive processing an implementation MUST also support the provisioning of received IP/UDP header information. When S-Labels are taken from platform label space, all that is required is to provision that receiving IP/UDP encapsulated DetNet MPLS packets is permitted. Once the IP/UDP header is stripped, the S-label uniquely identifies the app-flow. When S-Labels are not taken from platform label space, IP/UDP header information MUST be provisioned. The provisioned information MUST then be used to identify incoming app-

flows based on the combination of S-Label and incoming IP/UDP header. Normal receive processing, including PEOF can then take place.

#### 10. Security Considerations

The security considerations of DetNet in general are discussed in [I-D.ietf-detnet-architecture] and [I-D.sdt-detnet-security]. Other security considerations will be added in a future version of this draft.

#### 11. IANA Considerations

This document makes no IANA requests.

#### 12. Contributors

RFC7322 limits the number of authors listed on the front page of a draft to a maximum of 5, far fewer than the many individuals below who made important contributions to this draft. The editor wishes to thank and acknowledge each of the following authors for contributing text to this draft. See also Section 13.

Loa Andersson  
Huawei  
Email: loa@pi.nu

Yuanlong Jiang  
Huawei  
Email: jiangyuanlong@huawei.com

Norman Finn  
Huawei  
3101 Rio Way  
Spring Valley, CA 91977  
USA  
Email: norman.finn@mail01.huawei.com

Janos Farkas  
Ericsson  
Magyar Tudosok krt. 11.  
Budapest 1117  
Hungary  
Email: janos.farkas@ericsson.com

Carlos J. Bernardos  
Universidad Carlos III de Madrid  
Av. Universidad, 30  
Leganes, Madrid 28911

Spain  
Email: [cjbc@it.uc3m.es](mailto:cjbc@it.uc3m.es)

Tal Mizrahi  
Marvell  
6 Hamada st.  
Yokneam  
Israel  
Email: [talmi@marvell.com](mailto:talmi@marvell.com)

Lou Berger  
LabN Consulting, L.L.C.  
Email: [lberger@labn.net](mailto:lberger@labn.net)

Stewart Bryant  
Huawei Technologies  
Email: [stewart.bryant@gmail.com](mailto:stewart.bryant@gmail.com)

Mach Chen  
Huawei Technologies  
Email: [mach.chen@huawei.com](mailto:mach.chen@huawei.com)

Andrew G. Malis  
Huawei Technologies  
Email: [agmalis@gmail.com](mailto:agmalis@gmail.com)

### 13. Acknowledgements

The author(s) ACK and NACK.

The following people were part of the DetNet Data Plane Solution Design Team:

Jouni Korhonen

Janos Farkas

Norman Finn

Balazs Varga

Loa Andersson

Tal Mizrahi

David Mozes

Yuanlong Jiang

Andrew Malis

Carlos J. Bernardos

The DetNet chairs serving during the DetNet Data Plane Solution Design Team:

Lou Berger

Pat Thaler

Thanks for Stewart Bryant for his extensive review of the previous versions of the document.

## 14. References

### 14.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2211] Wroclawski, J., "Specification of the Controlled-Load Network Element Service", RFC 2211, DOI 10.17487/RFC2211, September 1997, <<https://www.rfc-editor.org/info/rfc2211>>.
- [RFC2212] Shenker, S., Partridge, C., and R. Guerin, "Specification of Guaranteed Quality of Service", RFC 2212, DOI 10.17487/RFC2212, September 1997, <<https://www.rfc-editor.org/info/rfc2212>>.
- [RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <<https://www.rfc-editor.org/info/rfc3031>>.
- [RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <<https://www.rfc-editor.org/info/rfc3032>>.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.

- [RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <<https://www.rfc-editor.org/info/rfc3270>>.
- [RFC3443] Agarwal, P. and B. Akyol, "Time To Live (TTL) Processing in Multi-Protocol Label Switching (MPLS) Networks", RFC 3443, DOI 10.17487/RFC3443, January 2003, <<https://www.rfc-editor.org/info/rfc3443>>.
- [RFC3473] Berger, L., Ed., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions", RFC 3473, DOI 10.17487/RFC3473, January 2003, <<https://www.rfc-editor.org/info/rfc3473>>.
- [RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, DOI 10.17487/RFC4206, October 2005, <<https://www.rfc-editor.org/info/rfc4206>>.
- [RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <<https://www.rfc-editor.org/info/rfc4385>>.
- [RFC5085] Nadeau, T., Ed. and C. Pignataro, Ed., "Pseudowire Virtual Circuit Connectivity Verification (VCCV): A Control Channel for Pseudowires", RFC 5085, DOI 10.17487/RFC5085, December 2007, <<https://www.rfc-editor.org/info/rfc5085>>.
- [RFC5129] Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, DOI 10.17487/RFC5129, January 2008, <<https://www.rfc-editor.org/info/rfc5129>>.
- [RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <<https://www.rfc-editor.org/info/rfc5462>>.
- [RFC7510] Xu, X., Sheth, N., Yong, L., Callon, R., and D. Black, "Encapsulating MPLS in UDP", RFC 7510, DOI 10.17487/RFC7510, April 2015, <<https://www.rfc-editor.org/info/rfc7510>>.

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

#### 14.2. Informative References

- [G.8275.1] International Telecommunication Union, "Precision time protocol telecom profile for phase/time synchronization with full timing support from the network", ITU-T G.8275.1/Y.1369.1 G.8275.1, June 2016, <<https://www.itu.int/rec/T-REC-G.8275.1/en>>.
- [G.8275.2] International Telecommunication Union, "Precision time protocol telecom profile for phase/time synchronization with partial timing support from the network", ITU-T G.8275.2/Y.1369.2 G.8275.2, June 2016, <<https://www.itu.int/rec/T-REC-G.8275.2/en>>.
- [I-D.ietf-detnet-architecture] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-11 (work in progress), February 2019.
- [I-D.ietf-detnet-dp-sol-ip] Korhonen, J., Varga, B., "DetNet IP Data Plane Encapsulation", 2018.
- [I-D.ietf-detnet-flow-information-model] Farkas, J., Varga, B., Cummings, R., and Y. Jiang, "DetNet Flow Information Model", draft-ietf-detnet-flow-information-model-03 (work in progress), March 2019.
- [I-D.ietf-pce-pcep-extension-for-pce-controller] Zhao, Q., Li, Z., Negi, M., and C. Zhou, "PCEP Procedures and Protocol Extensions for Using PCE as a Central Controller (PCECC) of LSPs", draft-ietf-pce-pcep-extension-for-pce-controller-01 (work in progress), February 2019.
- [I-D.ietf-spring-segment-routing-mpls] Bashandy, A., Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", draft-ietf-spring-segment-routing-mpls-18 (work in progress), December 2018.

- [I-D.sdt-detnet-security]  
Mizrahi, T., Grossman, E., Hacker, A., Das, S.,  
"Deterministic Networking (DetNet) Security  
Considerations, draft-sdt-detnet-security, work in  
progress", 2017.
- [IEEE1588]  
IEEE, "IEEE 1588 Standard for a Precision Clock  
Synchronization Protocol for Networked Measurement and  
Control Systems Version 2", 2008.
- [IEEE8021CB]  
Finn, N., "Draft Standard for Local and metropolitan area  
networks - Seamless Redundancy", IEEE P802.1CB  
/D2.1 P802.1CB, December 2015,  
<[http://www.ieee802.org/1/files/private/cb-drafts/  
d2/802-1CB-d2-1.pdf](http://www.ieee802.org/1/files/private/cb-drafts/d2/802-1CB-d2-1.pdf)>.
- [IEEE8021Q]  
IEEE 802.1, "Standard for Local and metropolitan area  
networks--Bridges and Bridged Networks (IEEE Std 802.1Q-  
2014)", 2014, <<http://standards.ieee.org/about/get/>>.
- [RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S.  
Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1  
Functional Specification", RFC 2205, DOI 10.17487/RFC2205,  
September 1997, <<https://www.rfc-editor.org/info/rfc2205>>.
- [RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black,  
"Definition of the Differentiated Services Field (DS  
Field) in the IPv4 and IPv6 Headers", RFC 2474,  
DOI 10.17487/RFC2474, December 1998,  
<<https://www.rfc-editor.org/info/rfc2474>>.
- [RFC3272] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and X.  
Xiao, "Overview and Principles of Internet Traffic  
Engineering", RFC 3272, DOI 10.17487/RFC3272, May 2002,  
<<https://www.rfc-editor.org/info/rfc3272>>.
- [RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation  
Edge-to-Edge (PWE3) Architecture", RFC 3985,  
DOI 10.17487/RFC3985, March 2005,  
<<https://www.rfc-editor.org/info/rfc3985>>.
- [RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron,  
"Encapsulation Methods for Transport of Ethernet over MPLS  
Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006,  
<<https://www.rfc-editor.org/info/rfc4448>>.



- [RFC4872] Lang, J., Ed., Rekhter, Y., Ed., and D. Papadimitriou, Ed., "RSVP-TE Extensions in Support of End-to-End Generalized Multi-Protocol Label Switching (GMPLS) Recovery", RFC 4872, DOI 10.17487/RFC4872, May 2007, <<https://www.rfc-editor.org/info/rfc4872>>.
- [RFC4873] Berger, L., Bryskin, I., Papadimitriou, D., and A. Farrel, "GMPLS Segment Recovery", RFC 4873, DOI 10.17487/RFC4873, May 2007, <<https://www.rfc-editor.org/info/rfc4873>>.
- [RFC4875] Aggarwal, R., Ed., Papadimitriou, D., Ed., and S. Yasukawa, Ed., "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, DOI 10.17487/RFC4875, May 2007, <<https://www.rfc-editor.org/info/rfc4875>>.
- [RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <<https://www.rfc-editor.org/info/rfc5440>>.
- [RFC5586] Bocci, M., Ed., Vigoureux, M., Ed., and S. Bryant, Ed., "MPLS Generic Associated Channel", RFC 5586, DOI 10.17487/RFC5586, June 2009, <<https://www.rfc-editor.org/info/rfc5586>>.
- [RFC5654] Niven-Jenkins, B., Ed., Brungard, D., Ed., Betts, M., Ed., Sprecher, N., and S. Ueno, "Requirements of an MPLS Transport Profile", RFC 5654, DOI 10.17487/RFC5654, September 2009, <<https://www.rfc-editor.org/info/rfc5654>>.
- [RFC5921] Bocci, M., Ed., Bryant, S., Ed., Frost, D., Ed., Levrau, L., and L. Berger, "A Framework for MPLS in Transport Networks", RFC 5921, DOI 10.17487/RFC5921, July 2010, <<https://www.rfc-editor.org/info/rfc5921>>.
- [RFC6003] Papadimitriou, D., "Ethernet Traffic Parameters", RFC 6003, DOI 10.17487/RFC6003, October 2010, <<https://www.rfc-editor.org/info/rfc6003>>.
- [RFC6006] Zhao, Q., Ed., King, D., Ed., Verhaeghe, F., Takeda, T., Ali, Z., and J. Meuric, "Extensions to the Path Computation Element Communication Protocol (PCEP) for Point-to-Multipoint Traffic Engineering Label Switched Paths", RFC 6006, DOI 10.17487/RFC6006, September 2010, <<https://www.rfc-editor.org/info/rfc6006>>.

- [RFC6073] Martini, L., Metz, C., Nadeau, T., Bocci, M., and M. Aissaoui, "Segmented Pseudowire", RFC 6073, DOI 10.17487/RFC6073, January 2011, <<https://www.rfc-editor.org/info/rfc6073>>.
- [RFC6387] Takacs, A., Berger, L., Caviglia, D., Fedyk, D., and J. Meuric, "GMPLS Asymmetric Bandwidth Bidirectional Label Switched Paths (LSPs)", RFC 6387, DOI 10.17487/RFC6387, September 2011, <<https://www.rfc-editor.org/info/rfc6387>>.
- [RFC6790] Kompella, K., Drake, J., Amante, S., Henderickx, W., and L. Yong, "The Use of Entropy Labels in MPLS Forwarding", RFC 6790, DOI 10.17487/RFC6790, November 2012, <<https://www.rfc-editor.org/info/rfc6790>>.
- [RFC7551] Zhang, F., Ed., Jing, R., and R. Gandhi, Ed., "RSVP-TE Extensions for Associated Bidirectional Label Switched Paths (LSPs)", RFC 7551, DOI 10.17487/RFC7551, May 2015, <<https://www.rfc-editor.org/info/rfc7551>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8169] Mirsky, G., Ruffini, S., Gray, E., Drake, J., Bryant, S., and A. Vainshtein, "Residence Time Measurement in MPLS Networks", RFC 8169, DOI 10.17487/RFC8169, May 2017, <<https://www.rfc-editor.org/info/rfc8169>>.

#### Appendix A. Example of DetNet Data Plane Operation

[Editor's note: Add a simplified example of DetNet data plane and how labels etc work in the case of MPLS-based PSN and utilizing PREOF. The figure is subject to change depending on the further DT decisions on the label handling..]

#### Authors' Addresses

Jouni Korhonen (editor)

Email: [jouni.nospam@gmail.com](mailto:jouni.nospam@gmail.com)

Balazs Varga (editor)  
Ericsson  
Magyar Tudosok krt. 11.  
Budapest 1117  
Hungary

Email: [balazs.a.varga@ericsson.com](mailto:balazs.a.varga@ericsson.com)

DetNet  
Internet-Draft  
Intended status: Standards Track  
Expires: September 12, 2019

J. Farkas  
B. Varga  
Ericsson  
R. Cummings  
National Instruments  
Y. Jiang  
Huawei Technologies Co., Ltd.  
March 11, 2019

DetNet Flow Information Model  
draft-ietf-detnet-flow-information-model-03

Abstract

This document describes flow and service information model for Deterministic Networking (DetNet). The DetNet service is provided either for a Layer 3 or a Layer 2 flow. This document provides DetNet flow and service information model both for Layer 3 and Layer 2 flows in an integrated fashion.

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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. ToDo list . . . . .	3
2. Introduction . . . . .	3
2.1. Goals . . . . .	5
2.2. Non Goals . . . . .	5
3. Conventions Used in This Document . . . . .	5
4. Terminology and Definitions . . . . .	5
5. Naming Conventions . . . . .	6
6. Service model . . . . .	6
6.1. Service overview . . . . .	6
6.2. Service parameters . . . . .	6
6.3. Reference Points . . . . .	8
6.4. Service scenarios . . . . .	9
7. End System and DetNet domain . . . . .	9
8. DetNet flows . . . . .	11
8.1. Identification and Specification of Flows . . . . .	11
8.1.1. IP flow Identification and Specification at UNI . . . . .	12
8.1.2. L2 Flow Identification and Specification at UNI . . . . .	12
8.1.3. DetNet Flow Identification and Specification . . . . .	13
8.2. Traffic Specification . . . . .	13
8.3. Flow Rank . . . . .	15
9. Source . . . . .	15
10. Destination . . . . .	16
11. Common Attributes of Source and Destination . . . . .	16
11.1. End System Interfaces . . . . .	16
11.2. Interface Capabilities . . . . .	16
11.3. User to Network Requirements . . . . .	17
12. Ingress . . . . .	18
13. Egress . . . . .	18
14. DetNet Domain . . . . .	18
14.1. DetNet Domain Capabilities . . . . .	19
15. Flow-status . . . . .	19
15.1. Status Info . . . . .	20
15.2. Interface Configuration . . . . .	21
15.3. Failed Interfaces . . . . .	21
16. Service Rank . . . . .	22
17. Service-status . . . . .	22
18. Summary . . . . .	22
19. IANA Considerations . . . . .	22
20. Security Considerations . . . . .	22
21. References . . . . .	22
21.1. Normative References . . . . .	23

21.2. Informative References . . . . .	23
Authors' Addresses . . . . .	24

## 1. ToDo list

These further actions are due on the draft:

- o Align with updated architecture and data plane documents (partly done).
- o App-flow parameters will not be defined in detail (add references only, e.g., 802.1Qcc). We focus on DetNet flows.
- o Clarification on relationship between DetNet flow model and DetNet service model.
- o Parameter set needs finalization, some re-org of the set may be needed.
- o Sort out which parameter belongs to DetNet flow model and which to DetNet service model.
- o Clarify relationship between App-flow and DetNet flow (N:1 vs 1:1).

## 2. Introduction

A Deterministic Networking (DetNet) service provides a capability to carry a unicast or a multicast data flow for an application with constrained requirements on network performance, e.g., low packet loss rate and/or latency. The DetNet service is provided either for a Layer 3 (L3) flow or a Layer 2 (L2) flow by an IP/MPLS network, see, e.g., [I-D.ietf-detnet-dp-sol-mpls]. Similarly, Time-Sensitive Networking (TSN) [IEEE8021TSN] can be used for L2 flows in a bridged network. DetNet and TSN have common architecture as expressed in [IETFDetNet] and [I-D.ietf-detnet-architecture]. DetNet service can be leveraged both by L3 and L2 flows, i.e., by DetNet L3 flows and DetNet L2 flows. Therefore, the DetNet flow and service information model provided by this document covers both DetNet L3 flows and DetNet L2 flows in an integrated fashion.

In a given network scenario three information models can be distinguished:

- o Flow models describe characteristics of data flows. These models describe in detail all relevant aspects of a flow that are needed to support the flow properly by the network between the source and the destination(s).

- o Service models describe characteristics of services being provided for data flows over a network. These models can be treated as a network operator independent information model.
- o Configuration models describe in detail the settings required on network nodes to serve a data flow properly.

Service and flow information models are used between the user and the network operator. Configuration information models are used between the management/control plane entity of the network and the network nodes. They are shown in Figure 1.

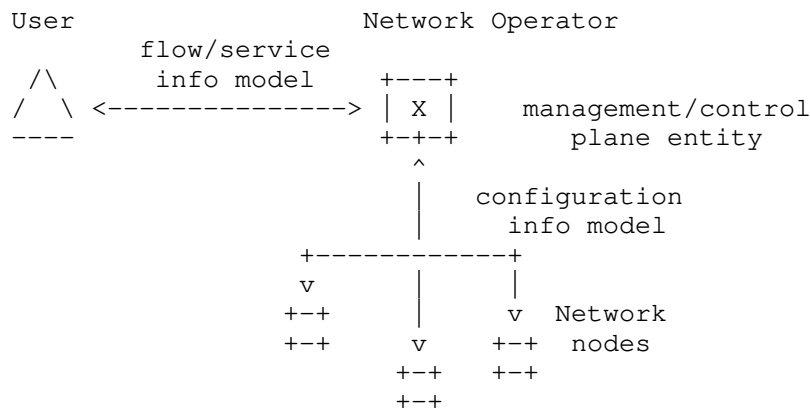


Figure 1: Usage of Information models (flow, service and configuration)

DetNet flow and service information model is based on [I-D.ietf-detnet-architecture] and on the data model specified by [IEEE8021Qcc]. Furthermore, the DetNet flow information model relies on the flow identification possibilities described in [IEEE8021CB], which is used by [IEEE8021Qcc] as well. In addition to TSN data model, [IEEE8021Qcc] also specifies configuration of TSN features (e.g., traffic scheduling specified by [IEEE8021Qbv]). Due to the common architecture and flow model, configuration features can be leveraged in certain deployment scenarios, e.g., when the network that provides the DetNet service includes both L3 and L2 network segments.

Based on the DetNet architecture [I-D.ietf-detnet-architecture] (see Section 4), this document (this revision) only considers the Centralized Network / Distributed User Model out of the models specified by [IEEE8021Qcc]. That is, there is a User-Network Interface (UNI) between an end system and a network. Furthermore, there is a central entity for the control of the network. For

instance, the central entity implements a Path Computation Element (PCE) for the calculation and establishment of paths needed for packet replication and elimination, if any.

## 2.1. Goals

As it is expressed in the Charter [IETFDetNet], the DetNet WG collaborates with IEEE 802.1 TSN in order to define a common architecture for both Layer 2 and Layer 3, which is beneficial for various reasons, e.g., in order to simplify implementations. The flow and service information models should be also common along those lines. As the TSN flow information/data model specified by [IEEE8021Qcc] is mature, the DetNet flow and service information models described in this document are based on [IEEE8021Qcc], which is an amendment to [IEEE8021Q].

This document intends to specify flow and service information models only.

## 2.2. Non Goals

This document (this revision) does not intend to specify either flow data model or DetNet configuration. From these aspects, the goals of this document differ from the goals of [IEEE8021Qcc], which also specifies data model and configuration of certain TSN features.

## 3. Conventions Used in This Document

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 [RFC2119].

The lowercase forms with an initial capital "Must", "Must Not", "Shall", "Shall Not", "Should", "Should Not", "May", and "Optional" in this document are to be interpreted in the sense defined in [RFC2119], but are used where the normative behavior is defined in documents published by SDOs other than the IETF.

## 4. Terminology and Definitions

This document uses the terminology established in Section 2 of the DetNet architecture document [I-D.ietf-detnet-architecture]. The DetNet <=> TSN dictionary of [I-D.ietf-detnet-architecture] is used to perform translation from [IEEE8021Qcc] to this document. Application level flows (app-flow) can be L2 or L3 flows, what may impact what header fields are use in order to identify a flow. DetNet flows are created by proper DetNet encapsulation of app-flow(s) (e.g., with added MPLS labels, etc.). In some scenarios App-



flow and DetNet flow looks similar on the wire (e.g., L3 App-flow over a DetNet IP network).

## 5. Naming Conventions

The following naming conventions were used for naming information model components in this document. It is recommended that extensions of the model use the same conventions.

- o Names SHOULD be descriptive.
- o Names MUST start with uppercase letters.
- o Composed names MUST use capital letters for the first letter of each component. All other letters are lowercase, even for acronyms. Exceptions are made for acronyms containing a mixture of lowercase and capital letters, such as IPv6. Examples are SourceMacAddress and DestinationIPv6Address.

## 6. Service model

### 6.1. Service overview

The DetNet service can be defined as a service that provides a capability to carry a unicast or a multicast data flow for an application with constrained requirements on network performance, e.g., low packet loss rate and/or latency.

The simplest DetNet service is to provide bridging over the DN domain (i.e., tunneling for L2), where the connected hosts are in the same broadcast (BC) domain. Forwarding over the DetNet domain is based on L2 (MAC) addresses (i.e. dst-MAC). Somewhat more sophisticated is DetNet Routing service that provides routing, so available only for L3 hosts that are in different BC domains. Forwarding over the DetNet domain is based on L3 (IP) addresses (i.e. dst-IP).

Figure 5. and Figure 8. in [I-D.ietf-detnet-architecture] show the DetNet service related reference points and main components.

### 6.2. Service parameters

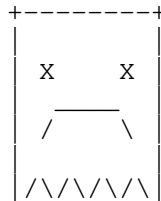
A DetNet network receives DetNet flows via a UNI as shown in Figure 5 in [I-D.ietf-detnet-architecture]. The DetNet network connects the UNIs via tunnels in order to provide DetNet service as shown in Figure 8 in [I-D.ietf-detnet-architecture].

The DetNet service attributes are the following:

- o Service type  
It is the flow type (L2 or L3) using the DetNet service.
- o Bandwidth  
It is the minimum bandwidth guaranteed for the DetNet service.
- o Delay parameters  
There are two delay parameters for a DetNet service:
  - \* Maximum latency, which is the maximum end-to-end one-way latency for the DetNet service.
  - \* Packet Delay Variation (PDV), which is the difference between the minimum and the maximum end-to-end one-way latency. The PDV parameter describes the maximum packet delay variation for the DetNet service. (Note that PDV is sometimes referred to as jitter.)
- o Loss parameters
  - \* The maximum Packet Loss Ratio (PLR) parameter describes the maximum packet loss ratio for the DetNet service between the edges of the DetNet network.
  - \* Some applications have special loss requirement. The maximum consecutive loss tolerance parameter describes the maximum number of consecutive packets whose loss can be tolerated. The maximum consecutive loss tolerance can be measured based on sequence number.
- o Maximum allowed misordering  
Maximum allowed misordering describes the tolerable maximum number of packets that can be received out of order. The maximum allowed misordering can be measured based on sequence number. The value zero for the maximum allowed misordering indicates that in order delivery is required, misordering cannot be tolerated.
- o Connectivity type  
Two connectivity types are distinguished: point-to-point (p2p) and point-to-multipoint (p2mp). Connectivity type p2mp is created by a transport layer function (e.g., p2mp LSP). (Note: mp2mp connectivity is a superposition of p2mp connections.)
- o Service rank  
Service rank provides the rank of a service instance relative to other services in the network. Rank is used by the network in case of network resource limitation scenarios.

### 6.3. Reference Points

From service model design perspective a fundamental question is the location of the service endpoints, i.e., where the service starts and ends.



To be ADDED  
404 Not Found

Figure 2: FIGURE Placeholder Reference Points

Note: Further discussion is needed based on data plane encapsulation results what reference points should be defined. Only some possible examples listed here:

- o App-flow endpoint: End system's internal reference point for the native data flow.
- o DetNet-UNI: UNI interface ("U") on a DetNet edge node.
- o DetNet-NNI: NNI interface ("N") between DetNet domains.

[[NOTE: Contributions are welcome whether we should define or distinguish internal reference point(s) for DetNet-aware end-systems as well. ]]

DetNet-UNI and DetNet-NNI are assumed in this document to be packet-based reference points and provide connectivity over the packet network and between domains. A DetNet-UNI may add networking technology specific encapsulation to the data flow in order to transport it over the network.

[[NOTE: Differences between the service over end-systems internal reference points and DetNet-UNI is for further discussions. For example, in-order delivery is expected in end system internal reference points, whereas it is considered optional over the DetNet-UNI. ]]

#### 6.4. Service scenarios

Using the above defined reference points, two major service scenarios can be identified:

- o End-to-End-Service: the service reaches out to final source or destination nodes, so it is an e2e service between application hosting devices (end systems).
- o DetNet-Service: the service connects networking islands, so it is a service between the borders of network domain(s).

[[NOTE: we may consider to define further scenarios based on the result of reference point related discussions. ]]

#### 7. End System and DetNet domain

Deterministic service is required by time/loss sensitive application(s) running on an end system during communication with its peer(s). Such a data exchange has various requirements on delay and/or loss parameters.

The DetNet architecture [I-D.ietf-detnet-architecture] distinguishes two kinds of end systems: Source and Destination. The same distinction is applied for the DetNet flow information model. In addition to the end systems interested in a flow, the status information of the flow is also important. Therefore, the DetNet flow information model relies on three high level groups:

- o Source: an end system capable of sourcing a DetNet flow. The Source information group includes elements that specify the Source for a single flow. This information group is applied from the user to the network.
- o Destination: an end system that is a destination of a DetNet flow. The Destination information group includes elements that specify the Destination for a single flow. This information group is applied from the user to the network.
- o Flow-Status: the status of a DetNet flow. The status information group includes elements that specify the status of the flow in the network. This information group is applied from the network to the user. This information group informs the user whether or not the flow is ready for use.

From service perspective two kinds of edge nodes can be distinguished: Ingress and Egress. In addition the technology of the DetNet domain and the status of the service are also important.

Therefore, the DetNet service information model relies on four high level groups:

- o **Ingress:** an edge system receiving a DetNet flow from a Source. The Ingress information group includes elements that specify the entry point for a single flow. This information group is applied from the network to the user.
- o **Egress:** an edge system sending traffic towards a Destination of a DetNet flow. The Egress information group includes elements that specify the egress point for a single flow. This information group is applied from the network to the user.
- o **DetNet Domain:** an administrative domain providing the DetNet service. The DetNet domain information group includes elements that specify the forwarding capabilities and methods for a single flow. This information group is applied within the network.
- o **Service-Status:** the status of a DetNet service. The status information group includes elements that specify the status of the service specific state of the network. This information group is applied from the network to the user. This information group informs the user whether or not the service is ready for use.

There are three operations for each flow with respect to a Source or a Destination (and an Ingress or an Egress):

- o **Join:** Source/Destination request to join the flow.
- o **Leave:** Source/Destination request to leave the flow.
- o **Modify:** Source/Destination request to change the flow.

Modify operation can be considered to address cases when a flow is slightly changed, e.g., only MaxPayloadSize (Section 8.2) has been changed. The advantage of having a Modify is that it allows to initiate a change of flow spec while leaving the current flow is operating until the change is accepted. If there is no linkage between the Join and the Leave, then in figuring out whether the new flow spec can be supported, the central entity has to assume that the resources committed to the current flow are in use. Via Modify the central entity knows that the resources supporting the current flow can be available for supporting the altered flow. Modify is considered to be an optional operation due to possible control-plane limitations.

As the DetNet UNI can provide service for both L3 and L2 app-flows, end systems may not need to implement the L3  $\Leftrightarrow$  L2 Transfer Function

specified by [IEEE8021CB] (see, e.g., subclause 6.3; see also subclause 46.1 in [IEEE8021Qcc]). A DetNet edge node may implement a function similar to the Transfer Function, see, e.g., the Svc Proxy in Figure 3 in [I-D.ietf-detnet-architecture].

## 8. DetNet flows

The app-flows leveraging DetNet service can be unicast or multicast data flows of an application with constrained requirements on network performance, e.g., low packet loss rate and/or latency. Flows can require different connectivity types: point-to-point (p2p) or point-to-multipoint (p2mp). The p2mp connectivity is created by a transport layer function (e.g., p2mp LSP) [I-D.ietf-detnet-dp-sol-mpls]. (Note that mp2mp connectivity is a superposition of p2mp connections.)

Many flows using DetNet service are periodic with fix packet size (i.e., Constant Bit Rate (CBR) flows), or periodic with variable packet size.

Delay and loss parameters are correlated because the effect of late delivery can result data loss for an application. However, not all applications require hard limits on both parameters (delay and loss). For example, some real-time applications allow graceful degradation if loss happens (e.g., sample-based processing, media distribution). Some others may require high-bandwidth connections that make the usage of techniques like packet replication economically challenging or even impossible. Some applications may not tolerate loss, but are not delay sensitive (e.g., bufferless sensors). Time/loss sensitive applications may have somewhat special requirements especially for loss (e.g., no loss in two consecutive communication cycles; very low outage time, etc.).

Flows have the following attributes:

- a. DataFlowSpecification (Section 8.1)
- b. TrafficSpecification (Section 8.2)
- c. FlowRank (Section 8.3)

Flow attributes are described in the following sections.

### 8.1. Identification and Specification of Flows

Identification options for DetNet flows at the UNI and within the DetNet domain are specified as follows; see Section 8.1.1 for DetNet

L3 flows (at UNI), Section 8.1.2 for DetNet L2 flows (at UNI) and Section 8.1.3 for DetNetwork flows (within the network).

#### 8.1.1. IP flow Identification and Specification at UNI

L3 data flows can be identified and specified by the following attributes:

- a. SourceIpAddress
- b. DestinationIpAddress
- c. IPv6FlowLabel
- d. Dscp
- e. Protocol
- f. SourcePort
- g. DestinationPort

#### 8.1.2. L2 Flow Identification and Specification at UNI

L2 data flows can be identified and specified by the following attributes:

- a. DestinationMacAddress
- b. SourceMacAddress
- c. Pcp
- d. VlanId
- e. EtherType

Note: The Multiple Stream Registration Protocol (MSRP) [IEEE8021Q] uses StreamID to match Talker registrations with their corresponding Listener registrations, i.e., to identify Streams (L2 TSN flows). The StreamID includes the following subcomponents:

- o A 48-bit MAC Address associated with the Talker sourcing the stream to the bridged network.
- o A 16-bit unsigned integer value, Unique ID, used to distinguish among multiple streams sourced by the same Talker.

### 8.1.3. DetNet Flow Identification and Specification

Identification of DetNet flows within the DetNet domain are used in the service information model. The attributes are specific to the forwarding paradigm within the DetNet domain. DetNetwork flows can be identified and specified by the following attributes:

- a. SourceIpAddress
- b. DestinationIpAddress
- c. IPv6FlowLabel
- d. DSCP
- e. Protocol
- f. SourcePort
- g. DestinationPort
- h. MplsLabel

### 8.2. Traffic Specification

TrafficSpecification specifies how the Source transmits packets for the flow. This is effectively the promise/request of the Source to the network. The network uses this traffic specification to allocate resources and adjust queue parameters in network nodes.

TrafficSpecification has the following attributes:

- a. Interval: the period of time in which the traffic specification cannot be exceeded.
- b. MaxPacketsPerInterval: the maximum number of packets that the Source will transmit in one Interval.
- c. MaxPayloadSize: the maximum payload size that the Source will transmit.

[[NOTE (to be removed from a future revision): These attributes can be used to describe any type of traffic (e.g., CBR, VBR, etc.) and can be used during resource allocation to represent worst case scenarios. Further optional attributes can be considered to achieve more efficient resource allocation. Such optional attributes might be worth for flows with soft requirements (i.e., the flow is only loss sensitive or only delay sensitive, but not both delay-and-loss



sensitive). Possible options how to extend TrafficSpecification attributes is for further discussion. Identified options are described in the following notes.]]

[[NOTE1: Based on the already defined attributes the most similar additional attributes for VBR type flows can be defined as follows:

- o AveragePacketsPerInterval: the average number of packets that the Source will transmit in one Interval.
- o AveragePayloadSize: the average payload size that the Source will transmit.

]]

[[NOTE2: another alternative to deal better with various traffic types can rely on [RFC6003], which describes the support of Metro Ethernet Forum (MEF) Ethernet traffic parameters for using for resource reservation purposes. Such a Bandwidth Profile can be also adapted to describe the set of traffic parameters for a Detnet flow. Committed Rate indicates the rate at which traffic commits to be sent by the source (described in terms of the CIR (Committed Information Rate) and CBS (Committed Burst Size) attributes.) Excess Rate indicates the extent by which the traffic sent by the source exceeds the committed rate. The Excess Rate is described in terms of the EIR (Excess Information Rate) and EBS (Excess Burst Size) attributes. ]]

[[NOTE3: a third alternative is to define application based traffic models such as [GPP22885] defines periodic and event-driven traffic model, and 5G PPP work defines traffic model for MTC (Machine Type Communication) use cases [I-D.ietf-detnet-use-cases]. Periodic traffic type is usually for status update between devices or devices transmit status report to a central unit in regular basis. TrafficPeriod, defines the period of the status update message. DataSize, defines the data size of the message which is constant. 3GPP also defines approximately-periodic transmission with variations on period and uncertainty in the time arrival of the packets. Event-triggered traffic type corresponds traffic being triggered by an MTC device event. MinIntervalBetweenEvent, defines the minimum interval between two events. Event-triggered transmission will not happen all the time, whenever an alert is sent, it waits until the issue being solved to be able to send another alert. MaxPacketPerEvent, defines the max number of packets within one message. ]]

### 8.3. Flow Rank

FlowRank provides the rank of this flow relative to other flows in the network. This rank is used to determine success/failure of flow establishment. Rank (boolean) is used by the network to decide which flows can and cannot exist when network resources reach their limit. Rank is used to help to determine which flows can be dropped (i.e., removed from node configuration) if a port of a node becomes oversubscribed (e.g., due to network reconfiguration). The true value is more important than the false value (i.e., flows with false are dropped first).

## 9. Source

The Source object specifies:

- o The behavior of the Source for the flow (how/when the Source transmits).
- o The requirements of the Source from the network.
- o The capabilities of the interface(s) of the Source.

The Source object includes the following attributes:

- a. DataFlowSpecification (Section 8.1)
- b. TrafficSpecification (Section 8.2)
- c. FlowRank (Section 8.3)
- d. EndSystemInterfaces (Section 11.1)
- e. InterfaceCapabilities (Section 11.2)
- f. UserToNetworkRequirements (Section 11.3)

For the join operation, the DataFlowSpecification, FlowRank, EndSystemInterfaces, and TrafficSpecification SHALL be included within the Source. For the join operation, the UserToNetworkRequirements and InterfaceCapabilities groups MAY be included within the Source.

For the leave operation, the DataFlowSpecification and EndSystemInterfaces SHALL be included within the Source.

For the modify operation, the same object SHALL and MAY included as for the join operation.

## 10. Destination

The Destination object includes the following attributes:

- a. DataFlowSpecification (Section 8.1)
- b. EndSystemInterfaces (Section 11.1)
- c. InterfaceCapabilities (Section 11.2)
- d. UserToNetworkRequirements (Section 11.3)

For the join operation, the DataFlowSpecification and EndSystemInterfaces SHALL be included within the Destination. For the join operation, the UserToNetworkRequirements and InterfaceCapabilities groups MAY be included within the Destination.

For the leave operation, the DataFlowSpecification and EndSystemInterfaces SHALL be included within the Destination.

For the modify operation, the same object SHALL and MAY included as for the join operation.

[[NOTE (to be removed from a future revision): Adding a general EndpointRank? That would define the endpoint importance (source, destination). It is only partly covered by FlowRank ... For example, it could distinguish the importance of Destinations if the flow cannot be provided for all Destinations.]]

## 11. Common Attributes of Source and Destination

Source and Destination end systems have the following common attributes in addition to DataFlowSpecification (Section 8.1).

### 11.1. End System Interfaces

EndSystemInterfaces is a list of identifiers, one for each physical interface (port) in the end system acting as a Source or Destination. An interface is identified by an IP or a MAC address.

EndSystemInterfaces can refer also to logical sub-Interfaces if supported by the end system, e.g., based on IfIndex parameter.

### 11.2. Interface Capabilities

InterfaceCapabilities specifies the network capabilities of all interfaces (ports) contained in the EndSystemInterfaces object (Section 11.1). These capabilities may be configured via the

InterfaceConfiguration object (Section 15.2) of the Status object (Section 15).

Note that an end system may have multiple interfaces with different network capabilities. In this case, each interface should be specified in a distinct top-level Source or Destination object (i.e., one entry in EndSystemInterfaces (Section 11.1)). Use of multiple entries in EndSystemInterfaces is intended for network capabilities that span multiple interfaces (e.g., packet replication and elimination).";.

InterfaceCapabilities attributes:

- a. SubInterfaceCapable (sub-interface capable)
- b. PREF-Capable (packet replication and elimination capable)
- c. POF-Capable (packet ordering capable)

[[NOTE (to be removed from a future revision): InterfaceCapabilities attributes are to be defined. For information, [IEEE8021Qcc] specifies the following attributes:

- o VlanTagCapable (Customer VLAN Tag capable)
- o CB-Capable (frame replication and elimination capable)
- o CB-StreamIdentTypeList (a list of the optional Stream Identification types supported by the interface as specified in [IEEE8021CB].)
- o CB-SequenceTypeList (a list of the optional Sequence Encode/Decode types supported by the interface as specified in [IEEE8021CB].)

]]

### 11.3. User to Network Requirements

UserToNetworkRequirements specifies user requirements for the flow, such as latency and reliability.

The UserToNetworkRequirements object includes the following attributes:

- a. MaxLatency

MaxLatency is the maximum latency from Source to Destination(s) for a single packet of the flow. MaxLatency is specified as an integer

number of nanoseconds. When this requirement is specified by the Source, it must be satisfied for all Destinations. When this requirement is specified by a Destination, it must be satisfied for that particular Destination only. If the UserToNetworkRequirements group is not provided within the Source or Destination object, then value zero SHALL be used for this element. Value zero represents a special use for the maximum latency requirement. Value zero locks-down the initial latency that the network provides in the AccumulatedLatency parameter of the Status object (Section 15) after the successful configuration of the flow, such that any subsequent increase in the latency beyond that initial value causes the flow to fail.

[[NOTE-1 (to be removed from a future revision): Should we add a parameter to specify the maximum packet loss rate that can be tolerated for the flow?]]

[[NOTE-2 (to be removed from a future revision): TrafficSpecification (Section 8.2) specifies the Peak Information Rate (PIR) of the flow, which is a kind of user requirement to the network. Should we add Committed Information Rate (CIR), i.e., the minimum rate the user requests to be guaranteed for the flow by the network?]]

## 12. Ingress

Placeholder ...

## 13. Egress

Placeholder ...

## 14. DetNet Domain

The DetNet Domain may change the encapsulation of a DetNet L2 or L3 flow at the UNI. That impacts not only how a flow can be recognised inside the DetNet domain but also the resource reservation calculations.

The DetNet Domain object specifies:

- o The behavior of the flow (how/when it is transmitted).
- o The requirements of the flow from the network.
- o The capabilities of the DetNet domain.

The DetNet domain object includes the following attributes:

- a. DataFlowSpecification (Section 8.1)
- b. TrafficSpecification (Section 8.2)
- c. ServiceRank (Section 16)
- d. DetnetDomainCapabilities (Section 14.1)
- e. UserToNetworkRequirements (Section 11.3)

#### 14.1. DetNet Domain Capabilities

DetnetDomainCapabilities specifies the network capabilities, which can be used to provide DetNet service. DetNet Edge nodes may change the encapsulation of a flow according to the data plane used inside the DetNet domain.

DetnetDomainCapabilities object includes the following attributes:

- a. EncapsulationFormat (data plane specific encapsulation)
- b. PREF-Capable (packet replication and elimination capable)

#### 15. Flow-status

The FlowStatus object is provided by the network each Source and Destination of the flow. The Status object provides the status of the flow with respect to the establishment of the flow by the network. The Status object is delivered via the corresponding UNI to each Source and Destination end system of the flow. The Status is distinct for each Source or Destination because the AccumulatedLatency and InterfaceConfiguration objects are distinct, see below.

The Status object SHALL include the attributes a), b), c); and MAY include attributes d), e):

- a. DataFlowSpecification (Section 8.1)
- b. StatusInfo (Section 15.1)
- c. AccumulatedLatency (this section below)
- d. InterfaceConfiguration (Section 15.2)
- e. FailedInterfaces (Section 15.3)

DataFlowSpecification identifies the flow for which status is provided. DataFlowSpecification is described in (Section 8.1) If the Status object is provided without a Source or Destination object in a protocol message via a UNI, then the DataFlowSpecification object SHALL be included within the Status object for both join and leave operations. If the Status object immediately follows a Source or Destination object in the protocol message, then the DataFlowSpecification object is obtained from the Source/Destination object, and therefore DataFlowSpecification is not required within the Status object.

AccumulatedLatency provides the worst-case latency that a single packet of the flow can encounter along its current path(s) in the network. When provided to a Source, AccumulatedLatency is the worst-case latency for all Destinations (worst path). AccumulatedLatency is specified as an integer number of nanoseconds. Latency is measured using the time at which the data frame's message timestamp point passes the reference plane marking the boundary between the network media and PHY. The message timestamp point is specified by IEEE Std 802.1AS [IEEE8021AS] for various media. For a successful Status, the network returns a value less than or equal to the MaxLatency of the UserToNetworkRequirements (Section 11.3). If the NumReplicationTrees of the UserToNetworkRequirements (Section 11.3) is one, then the AccumulatedLatency SHALL provide the worst latency for the current path from the Source to each Destination. If the path is changed (e.g., due to rerouting), then the AccumulatedLatency changes accordingly. If the NumReplicationTrees of the UserToNetworkRequirements (Section 11.3) is greater than one, AccumulatedLatency SHALL provide the worst latency for all paths in use from the Source to each Destination.

#### 15.1. Status Info

StatusInfo provides information regarding the status of a flow's configuration in the network.

The StatusInfo object MAY include the following attributes:

- a. SourceStatus is an enumeration for the status of the flow's Source:
  - \* None: no Source
  - \* Ready: Source is ready
  - \* Failed: Source failed

- b. `DestinationStatus` is an enumeration for the status of the flow's Destinations:
- \* `None`: no Destination
  - \* `Ready`: all Destinations are ready
  - \* `PartialFailed`: One or more Destinations ready, and one or more Listeners failed. The flow can be used if the Source is Ready.
  - \* `Failed`: All Destinations failed.
- c. `FailureCode`: A non-zero code that specifies the problem if the flow encounters a failure (e.g., packet replication and elimination is requested but not possible, or `SourceStatus` is `Failed`, or `DestinationStatus` is `Failed`, or `DestinationStatus` is `PartialFailed`).

[[NOTE (to be removed from a future revision): FailureCodes to be defined for DetNet. Table 46-1 of [IEEE8021Qcc] describes TSN failure codes.]]

#### 15.2. Interface Configuration

`InterfaceConfiguration` provides information about of interfaces in the Source/Destination. This configuration related information assists the network in meeting the requirements of the flow. The `InterfaceConfiguration` object is according to the capabilities of the interface. `InterfaceConfiguration` can be distinct for each Source or Destination of each flow. If the `InterfaceConfiguration` object is not provided within the Status object, then the network SHALL assume zero elements as the default (no interface configuration).

The `InterfaceConfiguration` object MAY include one or more the following attributes:

- a. MAC or IP Address to identify the interface
- b. `DataFlowSpecification` (Section 8.1)

#### 15.3. Failed Interfaces

`FailedInterfaces` provides a list of one or more physical interfaces (ports) in the failed node when a failure occurs in the network.

The `FailedInterface` object includes the following attributes:



- a. MAC or IP Address to identify the interface
- b. InterfaceName

InterfaceName is the name of the interface (port) within the node. This interface name SHALL be persistent, and unique within the node.

#### 16. Service Rank

ServiceRank provides the rank of this service instance relative to other services in the network. This rank is used to determine success/failure of service instance establishment. Rank (boolean) is used by the network to decide which services can and cannot exist when network resources reach their limit. Rank is used to help to determine which services can be dropped (i.e., removed from node configuration) if a port of a node becomes oversubscribed (e.g., due to network reconfiguration). The true value is more important than the false value (i.e., services with false are dropped first).

[[NOTE: relationship between ServiceRank and FlowRank needs further discussions. A 1:N relationship is assumed (a service instance can serv multiple flows). This sub-section is considered to move to the service related sections. ]]

#### 17. Service-status

Placeholder ...

#### 18. Summary

This document describes DetNet flow information model both for DetNet L3 flows and DetNet L2 flows based on the TSN data model specified by [IEEE8021Qcc]. This revision is extended with DetNet specific flow information model elements.

#### 19. IANA Considerations

N/A.

#### 20. Security Considerations

N/A.

#### 21. References

## 21.1. Normative References

- [I-D.ietf-detnet-architecture]  
Finn, N., Thubert, P., Varga, B., and J. Farkas,  
"Deterministic Networking Architecture", draft-ietf-  
detnet-architecture-11 (work in progress), February 2019.
- [I-D.ietf-detnet-dp-sol-mpls]  
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane  
Encapsulation", draft-ietf-detnet-dp-sol-mpls-01 (work in  
progress), October 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6003] Papadimitriou, D., "Ethernet Traffic Parameters",  
RFC 6003, DOI 10.17487/RFC6003, October 2010,  
<<https://www.rfc-editor.org/info/rfc6003>>.

## 21.2. Informative References

- [GPP22885]  
3GPP, "Study on LTE support for Vehicle-to-Everything  
(V2X) services",  
<<http://www.3gpp.org/DynaReport/22885.html>>.
- [I-D.ietf-detnet-use-cases]  
Grossman, E., "Deterministic Networking Use Cases", draft-  
ietf-detnet-use-cases-20 (work in progress), December  
2018.
- [IEEE8021AS]  
IEEE 802.1, "IEEE 802.1AS-2011: IEEE Standard for Local  
and metropolitan area networks - Timing and  
Synchronization for Time-Sensitive Applications in Bridged  
Local Area Networks", 2011,  
<[http://standards.ieee.org/getieee802/  
download/802.1AS-2011.pdf](http://standards.ieee.org/getieee802/download/802.1AS-2011.pdf)>.
- [IEEE8021CB]  
IEEE 802.1, "IEEE P802.1CB: IEEE Draft Standard for Local  
and metropolitan area networks - Frame Replication and  
Elimination for Reliability", 2017,  
<<http://www.ieee802.org/1/pages/802.1cb.html>>.

## [IEEE8021Q]

IEEE 802.1, "IEEE 802.1Q-2014: IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks", 2014, <<http://standards.ieee.org/getieee802/download/802-1Q-2014.pdf>>.

## [IEEE8021Qbv]

IEEE 802.1, "IEEE 802.1Qbv-2015: IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks -- Amendment 25: Enhancements for Scheduled Traffic", 2015, <<https://standards.ieee.org/findstds/standard/802.1Qbv-2015.html>>.

## [IEEE8021Qcc]

IEEE 802.1, "IEEE P802.1Qcc-2015: IEEE Draft Standard for Local and metropolitan area networks - Bridges and Bridged Networks -- Amendment: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements", 2017, <<http://www.ieee802.org/1/pages/802.1cc.html>>.

## [IEEE8021TSN]

IEEE 802.1, "IEEE 802.1 Time-Sensitive Networking (TSN) Task Group", <<http://www.ieee802.org/1/pages/tsn.html>>.

## [IETFDetNet]

IETF, "IETF Deterministic Networking (DetNet) Working Group", <<https://datatracker.ietf.org/wg/detnet/charter/>>.

## Authors' Addresses

Janos Farkas  
Ericsson  
Magyar tudosok korutja 11  
Budapest 1117  
Hungary

Email: [janos.farkas@ericsson.com](mailto:janos.farkas@ericsson.com)

Balazs Varga  
Ericsson  
Magyar tudosok korutja 11  
Budapest 1117  
Hungary

Email: [balazs.a.varga@ericsson.com](mailto:balazs.a.varga@ericsson.com)

Rodney Cummings  
National Instruments  
11500 N. Mopac Expwy  
Bldg. C  
Austin, TX 78759-3504  
USA

Email: [rodney.cummings@ni.com](mailto:rodney.cummings@ni.com)

Yuanlong Jiang  
Huawei Technologies Co., Ltd.  
Bantian, Longgang district  
Shenzhen 518129  
China

Email: [jiangyuanlong@huawei.com](mailto:jiangyuanlong@huawei.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: July 18, 2019

X. Geng  
M. Chen  
Huawei Technologies  
Z. Li  
China Mobile  
R. Rahman  
Cisco Systems  
January 14, 2019

Deterministic Networking (DetNet) Topology YANG Model  
draft-ietf-detnet-topology-yang-00

Abstract

This document defines a YANG data model for Deterministic Networking (DetNet) topology discovery and capability configuration.

The YANG module defined in this document conforms to the Network Management Datastore Architecture (NMDA).

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 RFC 2119 [RFC2119].

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 July 18, 2019.

## Copyright Notice

Copyright (c) 2019 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
2. Terminologies . . . . .	3
3. DetNet Topology Model . . . . .	3
3.1. DetNet Node Attributes . . . . .	4
3.2. DetNet Link Attributes . . . . .	4
3.3. DetNet Link Terminate Point Attributes . . . . .	5
4. DetNet Topology YANG Structure . . . . .	7
5. DetNet Topology YANG Model . . . . .	9
6. Open Issues . . . . .	14
7. IANA Considerations . . . . .	15
8. Security Considerations . . . . .	15
9. Acknowledgements . . . . .	15
10. References . . . . .	15
10.1. Normative References . . . . .	15
10.2. Informative References . . . . .	16
Authors' Addresses . . . . .	18

## 1. Introduction

Deterministic Networking (DetNet) [I-D.ietf-detnet-architecture] is defined to provide high-quality network service with extremely low packet loss rate, bounded low latency and jitter.

DetNet YANG [RFC7950] [RFC6991] models are used for DetNet service configuration, QoS configuration and topology discovery. DetNet service and QoS configuration models are defined in [I-D.ietf-detnet-yang]. This document defines DetNet topology model that can be used for DetNet topology/capability discovery and device configuration. DetNet topology model is an augmentation of the ietf-te-topology model [I-D.ietf-teas-yang-te-topo].

## 2. Terminologies

This document uses the terminologies defined in [I-D.ietf-detnet-architecture].

## 3. DetNet Topology Model

A DetNet topology is composed of a set of DetNet nodes and DetNet links. DetNet nodes represent the network devices that can transport DetNet services, which are connected by DetNet links. A DetNet Link Terminate Point (LTP) is the connection point between a DetNet node and a DetNet link, which represents the port or interface of a DetNet node. The concept of DetNet node/link/LTP are similar as TE node/link/LTP which are defined in [I-D.ietf-teas-yang-te-topo].

Figure 1 shows a simple DetNet topology: A is a DetNet node, B is DetNet a LTP, and C is a DetNet link.

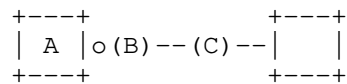


Figure 1. An example of DetNet Topology

DetNet topology model (ietf-detnet-topology) augments ietf-te-topology model [I-D.ietf-teas-yang-te-topo] to cover the following groups of attributes, which are necessary for supporting DetNet congestion protection and service protection:

- o Bandwidth related attributes (e.g., bandwidth reserved for DetNet);
- o Buffer/queue management related attributes (e.g., queue management parameters, etc.);
- o PREOF (Packet Replication, Elimination and Ordering Function) capabilities and parameters (e.g., maximum out-of-order packets, etc.);
- o Delay related attributes (e.g., node processing delay, queuing delay, link delay, etc.);

The above attributes are categorized into three types: node attributes, link attributes and LTP attributes. The detailed descriptions and model definitions are specified in section 3.1, 3.2 and 3.3, respectively.

### 3.1. DetNet Node Attributes

Section 4.3 of [I-D.finn-detnet-bounded-latency] gives a DetNet time model, which defines that the delay within a node includes five parts: processing delay, regulation delay, queuing delay, output delay and preemption delay. The processing delay, queuing delay and regulation delay are variable in general, but for DetNet, these delays should be bounded, which is the basic assumption of deterministic networking. These bounded delay parameters are necessary to perform DetNet path computation. Among this delay attributes, processing delay and regulation delay are node relevant, and the queuing delay is LTP relevant. In addition, in order to simplify the model and implementation, the processing delay and regulation delay are combined as processing delay, and the preemption delay is included in queuing delay. [Editor notes: more comments and inputs need here].

For the DetNet node attributes, the following variables are introduced:

- o Maximum DetNet packet processing delay
- o Minimum DetNet packet processing delay
- o Maximum DetNet packet processing delay variation

The modeling structure is shown below:

```
augment /nw:networks/nw:network/nw:node/tet:te/tet:te-node-attributes:
  +--rw detnet-node-attributes
    +--rw minimum-packet-processing-delay?      uint32
    +--rw maximum-packet-processing-delay?      uint32
    +--rw maximum-packet-processing-delay-variation?  uint32
```

### 3.2. DetNet Link Attributes

DetNet link attributes include link delay and link bandwidth for DetNet. This document introduces the following link related attributes:

- o Link delay: link delay is a constant that only depends on the physical connection. It has been defined in ietf-te-topology [I-D.ietf-teas-yang-te-topo], and DetNet can reuse it directly.
- o Maximum DetNet reservable bandwidth: the maximum reservable bandwidth that is allocated to DetNet. For a 10G link, if 50% of the bandwidth is allocated to DetNet, then the maximum DetNet



reservable bandwidth is 5G. That means there are 5G bandwidth that can be used by DetNet flows.

- o Reserved DetNet bandwidth: the bandwidth that has been reserved for DetNet flows.
- o Available DetNet bandwidth: the bandwidth that is available for new DetNet flows.

The DetNet link attributes are modeled within a link, and the YANG module structure is shown below:

```
augment /nw:networks/nw:network/nt:link/tet:te/tet:te-link-attributes:
  +--rw detnet-link-attributes
    +--rw maximum-reservable-bandwidth
      +--rw te-bandwidth
        +--rw (technology)?
          +--:(generic)
            +--rw generic?   te-bandwidth
      +--rw reserved-detnet-bandwidth
        +--rw te-bandwidth
          +--rw (technology)?
            +--:(generic)
              +--rw generic?   te-bandwidth
      +--rw available-detnet-bandwidth
        +--rw te-bandwidth
          +--rw (technology)?
            +--:(generic)
              +--rw generic?   te-bandwidth
```

### 3.3. DetNet Link Terminate Point Attributes

The concept of LTP is introduced in [I-D.ietf-teas-yang-te-topo], and this section introduces attributes for DetNet LTP.

PREOF (Packet Replication/Elimination/Ordering Function) is for DetNet service protection, which includes :

- o In-order delivery function: defined in Section 3.2.2.1 of [I-D.ietf-detnet-architecture]
- o Packet replication function: defined in Section 3.2.2.2 of [I-D.ietf-detnet-architecture]
- o Packet elimination function: defined in Section 3.2.2.3 of [I-D.ietf-detnet-architecture]

The above functions are modeled as a set of capabilities and relevant parameters, which are listed below:

- o in-order-capability: indicates whether a LTP has the in-order delivery capability.
- o maximum-number-of-out-of-order-packets: indicates the maximum number of out-of-order packets that an LTP can support, it depends on the reserved buffer size for packet reordering.
- o replication-capability: indicates whether a LTP has the packet replication capability.
- o elimination-capability: indicates whether a LTP has the packet elimination capability.

In addition, DetNet LTP also includes queuing management algorithms and queuing delay attributes. In the context of DetNet, the delay of queuing is bounded, and the bound depends on what queuing management method is used and how many buffers are allocated. More information can be found in [I-D.finn-detnet-bounded-latency]. Queuing related attributes are listed below:

- o queuing-algorithm-capabilities: it is modeled as a list that includes all queuing algorithms that a LTP supports.
- o detnet-queues: it's modeled as a list that includes all queues of a DetNet LTP. For each queue, it has the following attributes:
- o queue-identifier: an identifier of a queue. It could be an internal identifier that is only used within a node. Or it could be used by a centralized controller to specify in which specific queue a flow/packet is required to enter.
- o queue-buffer-size: the size of a queue with unit of bytes.
- o enabled-queuing-algorithm: indicates what queuing management algorithm is enabled.
- o maximum-queuing-delay: the maximum queuing delay that a packet will undergo when transmitted through the queue.
- o minimum-queuing-delay: the minimum queuing delay that a packet will undergo when transmitted through the queue.
- o maximum-queuing-delay-variation: the maximum queuing delay variation that a packet will undergo when transmitted the queue.

The DetNet LTP attributes are modeled within a LTP, the YANG module structure is shown below:

```
augment /nw:networks/nw:network/nw:node/nt:termination-point/tet:te:
  +--rw detnet-terminate-point-attributes
    +--rw elimination-capability?          boolean
    +--rw replication-capability?          boolean
    +--rw in-ordering-capability
      |
      +--rw in-ordering-capability?        boolean
      +--rw maximum-out-of-order-packets?  uint32
    +--rw queuing-algorithm-capabilities
      |
      +--rw credit-based-shaping?          boolean
      +--rw time-aware-shaping?           boolean
      +--rw cyclic-queuing-and-forwarding? boolean
      +--rw asynchronous-traffic-shaping?  boolean
    +--rw queues* [queue-identifier]
      +--rw queue-identifier                uint32
      +--rw queue-buffer-size?              uint32
      +--rw enabled-queuing-algorithm
        |
        +--rw credit-based-shaping?        boolean
        +--rw time-aware-shaping?          boolean
        +--rw cyclic-queuing-and-forwarding? boolean
        +--rw asynchronous-traffic-shaping? boolean
      +--rw minimum-queuing-delay?          uint32
      +--rw maximum-queuing-delay?          uint32
      +--rw maximum-queuing-delay-variation? uint32
```

#### 4. DetNet Topology YANG Structure

```

module: ietf-detnet-topology
augment /nw:networks/nw:network/nw:network-types/tet:te-topology:
  +--rw detnet-topology!
augment /nw:networks/nw:network/nw:node/tet:te/tet:te-node-attributes:
  +--rw detnet-node-attributes
    +--rw minimum-packet-processing-delay?          uint32
    +--rw maximum-packet-processing-delay?          uint32
    +--rw maximum-packet-processing-delay-variation? uint32
augment /nw:networks/nw:network/nt:link/tet:te/tet:te-link-attributes:
  +--rw detnet-link-attributes
    +--rw maximum-reservable-bandwidth
      +--rw te-bandwidth
        +--rw (technology)?
        +--:(generic)
          +--rw generic?   te-bandwidth
    +--rw reserved-detnet-bandwidth
      +--rw te-bandwidth
        +--rw (technology)?
        +--:(generic)
          +--rw generic?   te-bandwidth
    +--rw available-detnet-bandwidth
      +--rw te-bandwidth
        +--rw (technology)?
        +--:(generic)
          +--rw generic?   te-bandwidth
augment /nw:networks/nw:network/nw:node/nt:termination-point/tet:te:
  +--rw detnet-terminate-point-attributes
    +--rw elimination-capability?          boolean
    +--rw replication-capability?          boolean
    +--rw in-ordering-capability
      +--rw in-ordering-capability?        boolean
      +--rw maximum-out-of-order-packets?  uint32
    +--rw queuing-algorithm-capabilities
      +--rw credit-based-shaping?          boolean
      +--rw time-aware-shaping?            boolean
      +--rw cyclic-queuing-and-forwarding?  boolean
      +--rw asynchronous-traffic-shaping?   boolean
    +--rw queues* [queue-identifier]
      +--rw queue-identifier                uint32
      +--rw queue-buffer-size?              uint32
      +--rw enabled-queuing-algorithm
        +--rw credit-based-shaping?        boolean
        +--rw time-aware-shaping?          boolean
        +--rw cyclic-queuing-and-forwarding? boolean
        +--rw asynchronous-traffic-shaping? boolean
      +--rw minimum-queuing-delay?          uint32
      +--rw maximum-queuing-delay?          uint32
      +--rw maximum-queuing-delay-variation? uint32

```

## 5. DetNet Topology YANG Model

```
<CODE BEGINS> file "ietf-detnet-topology@20190114.yang"
module ietf-detnet-topology {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-detnet-topology";
  prefix "detnet-topology";

  import ietf-te-types {
    prefix "te-types";
  }

  import ietf-te-topology {
    prefix "tet";
  }

  import ietf-network {
    prefix "nw";
  }

  import ietf-network-topology {
    prefix "nt";
  }

  organization
    "IETF Deterministic Networking (DetNet) Working Group";

  contact
    "WG Web:  <http://tools.ietf.org/wg/detnet/>
    WG List:  <mailto:detnet@ietf.org>

    WG Chair: Lou Berger
               <mailto:lberger@labn.net>

               Janos Farkas
               <janos.farkas@ericsson.com>

    Editor:   Xuesong Geng
               <mailto:gengxuesong@huawei.com>

    Editor:   Mach Chen
               <mailto:mach.chen@huawei.com>

    Editor:   Zhenqiang Li
               <lizhenqiang@chinamobile.com>

    Editor:   Reshad Rahman
               <rrahman@cisco.com>";
```

```
description
  "This YANG module augments the 'ietf-te-topology'
  module with DetNet related capabilities and
  parameters.";

revision "2018-09-10" {
  description "Initial revision";
  reference "RFC XXXX: draft-geng-detnet-config-yang-05";
}

grouping detnet-queuing-algorithms {
  description
    "Relationship with IEEE 802.1 TSN YANG models is TBD.";
}

grouping detnet-node-attributes{
  description
    "DetNet node related attributes.";
  leaf minimum-packet-processing-delay{
    type uint32;
    description
      "Minimum packet processing delay
      in a node. The unit of the delay
      is microsecond(us) ";
  }
  leaf maximum-packet-processing-delay{
    type uint32;
    description
      "Maximum packet processing delay
      in a node. The unit of the delay
      is microsecond(us) ";
  }
  leaf maximum-packet-processing-delay-variation{
    type uint32;
    description
      "Maximum packet processing delay
      variation in a node. The unit of
      the delay variation is microsecond(us) ";
  }
}

grouping detnet-link-attributes{
  description
    "DetNet link related attributes.";

  container maximum-reservable-bandwidth{
    uses te-types:te-bandwidth;
```

```
    description
      "This container specifies the maximum bandwidth
       that is reserved for DetNet on this link.";
  }

  container reserved-detnet-bandwidth{
    uses te-types:te-bandwidth;
    description
      "This container specifies the bandwidth that has
       been reserved for DetNet on this link.";
  }
  container available-detnet-bandwidth{
    uses te-types:te-bandwidth;
    description
      "This container specifies the bandwidth that is
       available for new DetNet flows on this link.";
  }
}

grouping detnet-terminate-point-attributes{
  description
    "DetNet terminate point related attributes.";

  leaf elimination-capability{
    type boolean;
    description
      "Indicates whether a node is able to do packet
       elimination.";
    reference
      "Section 3.2.2.3 of
       draft-ietf-detnet-architecture";
  }

  leaf replication-capability{
    type boolean;
    description
      "Indicates whether a node is able to do packet
       replication.";
    reference
      "Section 3.2.2.2 of
       draft-ietf-detnet-architecture";
  }
}
container in-ordering-capability {
  description
    "Indicates the parameters needed for
     packet in-ordering.";
  reference
    "Section 3.2.2.1 of
```

```
    draft-ietf-detnet-architecture";

    leaf in-ordering-capability {
        type boolean;
        description
            "Indicates whether a node is able to do packet
             in-ordering.";
    }
    leaf maximum-out-of-order-packets {
        type uint32;
        description
            "The maximum number of out-of-order packets.";
    }
}

container queuing-algorithm-capabilities {
    description
        "All queuing algorithms that a LTP supports.";
    uses detnet-queuing-algorithms;
}

list queues {
    key "queue-identifier";
    description
        "A list of DetNet queues.";
    leaf queue-identifier {
        type uint32;
        description
            "The identifier of the queue.";
    }
    leaf queue-buffer-size {
        type uint32;
        description
            "The size of the queue with unit of bytes.";
    }
}

container enabled-queuing-algorithm {
    description
        "The queuing algorithms that are enabled on the queue.";
    uses detnet-queuing-algorithms;
}

leaf minimum-queuing-delay{
    type uint32;
    description
        "The minimum queuing delay of the queue.
         The unit of the delay is microsecond(us)";
}
```



```
leaf maximum-queuing-delay{
  type uint32;
  description
    "The maximum queuing delay of the queue.
    The unit of the delay is microsecond(us)";
}
leaf maximum-queuing-delay-variation{
  type uint32;
  description
    "The maximum queuing delay variation of the queue.
    The unit of the delay variation is microsecond(us)";
}
}

augment "/nw:networks/nw:network/nw:network-types/tet:te-topology"{
  description
    "Introduce new network type for TE topology.";
  container detnet-topology {
    presence "Indicates DetNet topology.";
    description
      "Its presence identifies the DetNet topology type";
  }
}

augment "/nw:networks/nw:network/nw:node/tet:te/"
  + "tet:te-node-attributes" {
  when "../..../nw:network-types/tet:te-topology/"
  + "detnet-topology:detnet-topology" {
    description
      "Augmentation parameters apply only for networks with
      DetNet topology type.";
  }
  description
    "Augmentation parameters apply for DetNet node attributes.";
  container detnet-node-attributes {
    description
      "Attributes for DetNet node.";
    uses detnet-node-attributes;
  }
}

augment "/nw:networks/nw:network/nt:link/tet:te/"
  + "tet:te-link-attributes" {
  when "../..../nw:network-types/tet:te-topology/"
  + "detnet-topology:detnet-topology" {
    description
      "Augmentation parameters apply only for networks with
```

```

        DetNet topology type.";
    }
    description
        "Augmentation parameters apply for DetNet link attributes.";
    container detnet-link-attributes {
        description
            "Attributes for DetNet link.";
        uses detnet-link-attributes;
    }
}

augment "/nw:networks/nw:network/nw:node/nt:termination-point/"
    + "tet:te" {
    when "../..../nw:network-types/tet:te-topology/"
        + "detnet-topology:detnet-topology" {
        description
            "Augmentation parameters apply only for networks with
            DetNet topology type.";
    }
    description
        "Augmentation parameters apply for DetNet
        link termination point.";
    container detnet-terminate-point-attributes {
        description
            "Attributes for DetNet link terminate point.";
        uses detnet-terminate-point-attributes;
    }
}
} //topology module

```

<CODE ENDS>

## 6. Open Issues

There are some open issues that are still under discussion:

- o The Relationship with 802.1 TSN YANG models is TBD. TSN YANG models include: P802.1Qcw, which defines TSN YANG for Qbv, Qbu, and Qci, and P802.1CBcv, which defines YANG for 802.1CB. The possible problem here is how to avoid possible overlap among yang models defined in IETF and IEEE. A common YANG model may be defined in the future to shared by both TSN and DetNet. More discussion are needed here.
- o How to support DetNet OAM is TBD.

These issues will be resolved in the following versions of the draft.

## 7. IANA Considerations

This document makes no request of IANA.

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

## 8. Security Considerations

<TBD>

## 9. Acknowledgements

## 10. References

### 10.1. Normative References

- [I-D.finn-detnet-bounded-latency]  
Finn, N., Boudec, J., Mohammadpour, E., Zhang, J., Varga, B., and J. Farkas, "DetNet Bounded Latency", draft-finn-detnet-bounded-latency-02 (work in progress), October 2018.
- [I-D.ietf-detnet-architecture]  
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-10 (work in progress), December 2018.
- [I-D.ietf-detnet-dp-sol-ip]  
Korhonen, J. and B. Varga, "DetNet IP Data Plane Encapsulation", draft-ietf-detnet-dp-sol-ip-01 (work in progress), October 2018.
- [I-D.ietf-detnet-dp-sol-mpls]  
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane Encapsulation", draft-ietf-detnet-dp-sol-mpls-01 (work in progress), October 2018.
- [I-D.ietf-detnet-flow-information-model]  
Farkas, J., Varga, B., Cummings, R., Jiang, Y., and Y. Zha, "DetNet Flow Information Model", draft-ietf-detnet-flow-information-model-02 (work in progress), October 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.

- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.

## 10.2. Informative References

- [I-D.geng-detnet-info-distribution]  
Geng, X., Chen, M., and Z. Li, "IGP-TE Extensions for DetNet Information Distribution", draft-geng-detnet-info-distribution-03 (work in progress), October 2018.
- [I-D.ietf-detnet-use-cases]  
Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-20 (work in progress), December 2018.
- [I-D.ietf-detnet-yang]  
Geng, X., Chen, M., Li, Z., and R. Rahman, "DetNet Configuration YANG Model", draft-ietf-detnet-yang-00 (work in progress), October 2018.
- [I-D.ietf-teas-yang-te]  
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-17 (work in progress), October 2018.
- [I-D.ietf-teas-yang-te-topo]  
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-18 (work in progress), June 2018.
- [I-D.thubert-tsvwg-detnet-transport]  
Thubert, P., "A Transport Layer for Deterministic Networks", draft-thubert-tsvwg-detnet-transport-01 (work in progress), October 2017.
- [I-D.varga-detnet-service-model]  
Varga, B. and J. Farkas, "DetNet Service Model", draft-varga-detnet-service-model-02 (work in progress), May 2017.

- [IEEE802.1CB]  
"IEEE, "Frame Replication and Elimination for Reliability (IEEE Draft P802.1CB)", 2017,  
<<http://www.ieee802.org/1/files/private/cb-drafts/>>.", 2016.
- [IEEE802.1Q-2014]  
"IEEE, "IEEE Std 802.1Q Bridges and Bridged Networks", 2014, <<http://ieeexplore.ieee.org/document/6991462/>>.", 2014.
- [IEEE802.1Qbu]  
"IEEE, "IEEE Std 802.1Qbu Bridges and Bridged Networks - Amendment 26: Frame Preemption", 2016,  
<<http://ieeexplore.ieee.org/document/7553415/>>.", 2016.
- [IEEE802.1Qbv]  
"IEEE, "IEEE Std 802.1Qbu Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic", 2015,  
<<http://ieeexplore.ieee.org/document/7572858/>>.", 2016.
- [IEEE802.1Qcc]  
"IEEE, "Stream Reservation Protocol (SRP) Enhancements and Performance Improvements (IEEE Draft P802.1Qcc)", 2017,  
<<http://www.ieee802.org/1/files/private/cc-drafts/>>.",
- [IEEE802.1Qch]  
"IEEE, "Cyclic Queuing and Forwarding (IEEE Draft P802.1Qch)", 2017,  
<<http://www.ieee802.org/1/files/private/ch-drafts/>>.", 2016.
- [IEEE802.1Qci]  
"IEEE, "Per-Stream Filtering and Policing (IEEE Draft P802.1Qci)", 2016,  
<<http://www.ieee802.org/1/files/private/ci-drafts/>>.", 2016.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.

- [RFC4875] Aggarwal, R., Ed., Papadimitriou, D., Ed., and S. Yasukawa, Ed., "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, DOI 10.17487/RFC4875, May 2007, <<https://www.rfc-editor.org/info/rfc4875>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.

## Authors' Addresses

Xuesong Geng  
Huawei Technologies

Email: [gengxuesong@huawei.com](mailto:gengxuesong@huawei.com)

Mach(Guoyi) Chen  
Huawei Technologies

Email: [mach.chen@huawei.com](mailto:mach.chen@huawei.com)

Zhenqiang Li  
China Mobile

Email: [lizhenqiang@chinamobile.com](mailto:lizhenqiang@chinamobile.com)

Reshad Rahman  
Cisco Systems

Email: [rrahman@cisco.com](mailto:rrahman@cisco.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: July 18, 2019

X. Geng  
M. Chen  
Huawei Technologies  
Z. Li  
China Mobile  
R. Rahman  
Cisco Systems  
January 14, 2019

Deterministic Networking (DetNet) Configuration YANG Model  
draft-ietf-detnet-yang-01

Abstract

This document contains the specification for Deterministic Networking flow configuration YANG Model. The model allows for provisioning of end-to-end DetNet service along the path without dependency on any signaling protocol.

The YANG module defined in this document conforms to the Network Management Datastore Architecture (NMDA).

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 RFC 2119 [RFC2119].

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 July 18, 2019.

## Copyright Notice

Copyright (c) 2019 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
2. Terminologies . . . . .	4
3. DetNet Configuration Model . . . . .	4
3.1. DetNet Service Proxy Configuration Attributes . . . . .	4
3.2. DetNet Service Layer Configuration Attributes . . . . .	5
3.3. DetNet Transport Layer Configuration Attributes . . . . .	7
4. DetNet Configuration YANG Structure . . . . .	8
5. DetNet Configuration YANG Model . . . . .	14
6. DetNet Configuration Model Classification . . . . .	31
6.1. Fully Distributed Configuration Model . . . . .	31
6.2. Fully Centralized Configuration Model . . . . .	31
6.3. Hybrid Configuration Model . . . . .	32
7. Open Issues . . . . .	33
8. IANA Considerations . . . . .	33
9. Security Considerations . . . . .	33
10. Acknowledgements . . . . .	34
11. References . . . . .	34
11.1. Normative References . . . . .	34
11.2. Informative References . . . . .	35
Authors' Addresses . . . . .	37

## 1. Introduction

Deterministic Networking (DetNet) [I-D.ietf-detnet-architecture] is defined to provide high-quality network service with extremely low packet loss rate, bounded low latency and jitter.

DetNet flow information is defined in [I-D.ietf-detnet-flow-information-model], and the DetNet models are categorized as:



- o Flow models: describe characteristics of data flows. These models describe in detail all relevant aspects of a flow that are needed to support the flow properly by the network between the source and the destination(s).
- o Service models: describe characteristics of services being provided for data flows over a network. These models can be treated as a network operator independent information model.
- o Configuration models: describe in detail the settings required on network nodes to serve a data flow properly. Service and flow information models are used between the user and the network operator. Configuration information models are used between the management/control plane entity of the network and the network nodes.

They are shown in the Figure 1.

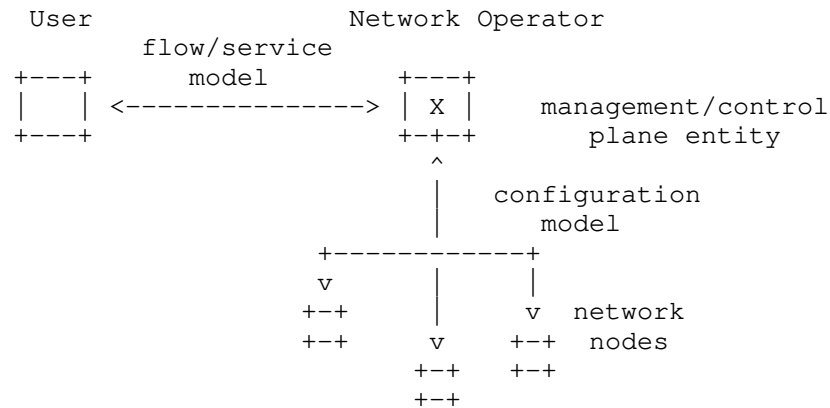


Figure 1. Three Information Models

DetNet YANG [RFC7950] [RFC6991] models include:

DetNet YANG [RFC7950] [RFC6991] models are used for DetNet service configurations, QoS configuration and topology discovery. DetNet topology model is defined in ietf-detnet-topology-yang. This document defines two YANG models, which are referred to as DetNet flow configuration model and DetNet transport QoS model. DetNet flow model is designed for DetNet flow path configuration and flow status reporting. DetNet transport QoS model is designed for QoS attributes configuration of transport tunnels to achieve end-to-end bounded latency and zero congestion loss.

## 2. Terminologies

This documents uses the terminologies defined in [I-D.ietf-detnet-architecture].

## 3. DetNet Configuration Model

DetNet flow configuration includes DetNet Service Proxy configuration, DetNet Service Layer configuration and DetNet Transport Layer configuration. The corresponding attributes used in different layers are defined in Section 3.1, 3.2, 3.3, respectively.

### 3.1. DetNet Service Proxy Configuration Attributes

DetNet service proxy is responsible for mapping between application flows and DetNet flows at the edge node (egress/ingress node). Where the application flows can be either layer 2 or layer 3 flows. To identify a flow at the User Network Interface (UNI), as defined in [I-D.ietf-detnet-flow-information-model], the following flow attributes are introduced:

- o DetNet L3 Flow Identification, refers to Section 7.1.1 of [I-D.ietf-detnet-flow-information-model]
- o DetNet L2 Flow Identification, refers to Section 7.1.2 of [I-D.ietf-detnet-flow-information-model]

DetNet service proxy can also do flow filtering and policing at the ingress to prevent the misbehaved flows from going into the network, which needs:

- o Traffic Specification, refers to Section 7.2 of [I-D.ietf-detnet-flow-information-model]

The YANG module structure is shown below:

```

+--rw client-flow* [flow-id]
|   +--rw flow-id                               uint32
|   +--rw (flow-type)?
|   |   +--:(l2-flow-identification)
|   |   |   +--rw source-mac-address?          yang:mac-address
|   |   |   +--rw destination-mac-address?     yang:mac-address
|   |   |   +--rw ethertype?                   eth:ethertype
|   |   |   +--rw vlan-id?                     uint16
|   |   |   +--rw pcp
|   |   +--:(l3-flow-identification)
|   |   |   +--rw (ip-flow-type)?
|   |   |   |   +--:(ipv4)
|   |   |   |   |   +--rw src-ipv4-address?    inet:ipv4-address
|   |   |   |   |   +--rw dest-ipv4-address?  inet:ipv4-address
|   |   |   |   |   +--rw dscp?               uint8
|   |   |   |   +--:(ipv6)
|   |   |   |   |   +--rw src-ipv6-address?    inet:ipv6-address
|   |   |   |   |   +--rw dest-ipv6-address?  inet:ipv6-address
|   |   |   |   |   +--rw traffic-class?      uint8
|   |   |   |   |   +--rw flow-label?         inet:ipv6-flow-label
|   |   |   +--rw source-port?                inet:port-number
|   |   |   +--rw destination-port?           inet:port-number
|   |   |   +--rw protocol?                   uint8
|   +--rw traffic-specification
|   |   +--rw interval?                       uint32
|   |   +--rw max-packets-per-interval?       uint32
|   |   +--rw max-payload-size?               uint32
|   |   +--rw average-packets-per-interval?   uint32
|   |   +--rw average-payload-size?           uint32

```

### 3.2. DetNet Service Layer Configuration Attributes

DetNet service functions, e.g., DetNet tunnel initialization/termination and service protection, are provided in DetNet service layer. To support these functions, the following service attributes need to be configured:

- o DetNet flow identification, refers to Section 7.1.3 of [I-D.ietf-detnet-flow-information-model].
- o Service function indication, indicates which service function will be invoked at a DetNet edge, relay node or end station. (DetNet tunnel initialization or termination are default functions in DetNet service layer, so there is no need for explicit indication.)
- o Flow Rank, refers to Section 7.3 of [I-D.ietf-detnet-flow-information-model].

- o Service Rank, refers to Section 7.4 of [I-D.ietf-detnet-flow-information-model].
- o Service decapsulation, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls]
- o Transport decapsulation, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls] and Section 3 of [I-D.ietf-detnet-dp-sol-ip]
- o Service encapsulation, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls]
- o Transport encapsulation, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls] and Section 3 of [I-D.ietf-detnet-dp-sol-ip]

The YANG module structure is shown below:

```

+--rw relay-node
  +--rw name?                string
  +--rw flow-rank
  +--rw service-rank
  +--rw in-segment* [in-segment-id]
    +--rw in-segment-id      uint32
    +--rw (flow-type)?
      +--:(IP)
        +--rw (ip-flow-type)?
          +--:(ipv4)
            +--rw src-ipv4-address?    inet:ipv4-address
            +--rw dest-ipv4-address?   inet:ipv4-address
            +--rw dscp?                 uint8
          +--:(ipv6)
            +--rw src-ipv6-address?    inet:ipv6-address
            +--rw dest-ipv6-address?   inet:ipv6-address
            +--rw traffic-class?       uint8
            +--rw flow-label?          inet:ipv6-flow-label
        +--rw source-port?             inet:port-number
        +--rw destination-port?        inet:port-number
        +--rw protocol?                uint8
      +--:(MPLS)
        +--rw service-label            uint32
    +--rw service-function?           service-function-type
  +--rw out-segment* [out-segment-id]
    +--rw out-segment-id              uint32
  +--rw detnet-service-encapsulation
    +--rw service-label                uint32
    +--rw control-word                  uint32

```

```

+--rw detnet-transport-encapsulation
+--rw (tunnel-type)?
+--:(IPv4)
+--rw ipv4-encapsulation
+--rw src-ipv4-address      inet:ipv4-address
+--rw dest-ipv4-address    inet:ipv4-address
+--rw protocol              uint8
+--rw ttl?                  uint8
+--rw dscp?                 uint8
+--:(IPv6)
+--rw ipv6-encapsulation
+--rw src-ipv6-address      inet:ipv6-address
+--rw dest-ipv6-address    inet:ipv6-address
+--rw next-header           uint8
+--rw traffic-class?       uint8
+--rw flow-label?          inet:ipv6-flow-label
+--rw hop-limit?           uint8
+--:(MPLS)
+--rw mpls-encapsulation
+--rw label-operations* [label-oper-id]
+--rw label-oper-id        uint32
+--rw (label-actions)?
+--:(label-push)
+--rw label-push
+--rw label                 uint32
+--rw s-bit?                boolean
+--rw tc-value?             uint8
+--rw ttl-value?           uint8
+--:(label-swap)
+--rw label-swap
+--rw out-label             uint32
+--rw ttl-action?          ttl-action-definition
+--rw interval?            uint32
+--rw max-packets-per-interval? uint32
+--rw max-payload-size?    uint32
+--rw average-packets-per-interval? uint32
+--rw average-payload-size? uint32

```

### 3.3. DetNet Transport Layer Configuration Attributes

As defined in [I-D.ietf-detnet-architecture], DetNet transport layer optionally provides congestion protection for DetNet flows over paths provided by the underlying network. Explicit route is another mechanism that is used by DetNet to avoid temporary interruptions caused by the convergence of routing or bridging protocols, and it is also implemented at the DetNet transport layer.

To support congestion protection and explicit route, the following transport layer related attributes are necessary:

- o Traffic Specification, refers to Section 7.2 of [I-D.ietf-detnet-flow-information-model]. It may used for bandwidth reservation, flow shaping, filtering and policing.
- o Explicit path, existing explicit route mechanisms can be reused. For example, if Segment Routing (SR) tunnel is used as the transport tunnel, the configuration is mainly at the ingress node of the transport layer; if the static MPLS tunnel is used as the transport tunnel, the configurations need to be at every transit node along the path; for pure IP based transport tunnel, it's similar to the static MPLS case.

The YANG module structure is shown below:

```

+--rw transit-node
|   +--rw interval?                               uint32
|   +--rw max-packets-per-interval?               uint32
|   +--rw max-payload-size?                       uint32
|   +--rw average-packets-per-interval?           uint32
|   +--rw average-payload-size?                   uint32

```

The parameters for DetNet transport QoS are defined in Section 5.

#### 4. DetNet Configuration YANG Structure

```

module: ietf-detnet-flow-config
+--rw detnet-flow
|   +--rw (detnet-node-role)?
|   |   +--:(transit-node)
|   |   |   +--rw transit-node
|   |   |   |   +--rw interval?                               uint32
|   |   |   |   +--rw max-packets-per-interval?               uint32
|   |   |   |   +--rw max-payload-size?                       uint32
|   |   |   |   +--rw average-packets-per-interval?           uint32
|   |   |   |   +--rw average-payload-size?                   uint32
|   |   |   +--:(relay-node)
|   |   |   +--rw relay-node
|   |   |   |   +--rw name?                                     string
|   |   |   |   +--rw flow-rank
|   |   |   |   +--rw service-rank
|   |   |   |   +--rw in-segment* [in-segment-id]
|   |   |   |   |   +--rw in-segment-id                       uint32
|   |   |   |   |   +--rw (flow-type)?
|   |   |   |   |   |   +--:(IP)
|   |   |   |   |   |   |   +--rw (ip-flow-type)?

```

```

+---:(ipv4)
|   +---rw src-ipv4-address?      inet:ipv4-address
|   +---rw dest-ipv4-address?     inet:ipv4-address
|   +---rw dscp?                  uint8
+---:(ipv6)
|   +---rw src-ipv6-address?      inet:ipv6-address
|   +---rw dest-ipv6-address?     inet:ipv6-address
|   +---rw traffic-class?         uint8
|   +---rw flow-label?            inet:ipv6-flow-label
+---rw source-port?               inet:port-number
+---rw destination-port?          inet:port-number
+---rw protocol?                  uint8
+---:(MPLS)
|   +---rw service-label          uint32
+---rw service-function?          service-function-type
+---rw out-segment* [out-segment-id]
|   +---rw out-segment-id          uint32
+---rw detnet-service-encapsulation
|   +---rw service-label          uint32
|   +---rw control-word           uint32
+---rw detnet-transport-encapsulation
+---rw (tunnel-type)?
|   +---:(IPv4)
|   |   +---rw ipv4-encapsulation
|   |   |   +---rw src-ipv4-address      inet:ipv4-address
|   |   |   +---rw dest-ipv4-address     inet:ipv4-address
|   |   |   +---rw protocol              uint8
|   |   |   +---rw ttl?                  uint8
|   |   |   +---rw dscp?                  uint8
|   |   +---:(IPv6)
|   |   |   +---rw ipv6-encapsulation
|   |   |   |   +---rw src-ipv6-address  inet:ipv6-address
|   |   |   |   +---rw dest-ipv6-address inet:ipv6-address
|   |   |   |   +---rw next-header       uint8
|   |   |   |   +---rw traffic-class?    uint8
|   |   |   |   +---rw flow-label?       inet:ipv6-flow-label
|   |   |   |   +---rw hop-limit?        uint8
|   |   +---:(MPLS)
|   |   |   +---rw mpls-encapsulation
|   |   |   |   +---rw label-operations* [label-oper-id]
|   |   |   |   |   +---rw label-oper-id  uint32
|   |   |   |   |   +---rw (label-actions)?
|   |   |   |   |   |   +---:(label-push)
|   |   |   |   |   |   |   +---rw label-push
|   |   |   |   |   |   |   |   +---rw label      uint32
|   |   |   |   |   |   |   |   +---rw s-bit?     boolean
|   |   |   |   |   |   |   |   +---rw tc-value?   uint8
|   |   |   |   |   |   |   |   +---rw ttl-value?  uint8

```

```

+---: (label-swap)
+---rw label-swap
+---rw out-label          uint32
+---rw ttl-action?       ttl-action-definition
+---rw interval?         uint32
+---rw max-packets-per-interval?    uint32
+---rw max-payload-size?    uint32
+---rw average-packets-per-interval? uint32
+---rw average-payload-size?    uint32
+---: (edge-node)
+---rw edge-node
+---rw client-flow* [flow-id]
+---rw flow-id          uint32
+---rw (flow-type)?
+---: (l2-flow-identification)
+---rw source-mac-address?    yang:mac-address
+---rw destination-mac-address? yang:mac-address
+---rw ethertype?            eth:ethertype
+---rw vlan-id?              uint16
+---rw pcp
+---: (l3-flow-identification)
+---rw (ip-flow-type)?
+---: (ipv4)
+---rw src-ipv4-address?      inet:ipv4-address
+---rw dest-ipv4-address?     inet:ipv4-address
+---rw dscp?                  uint8
+---: (ipv6)
+---rw src-ipv6-address?      inet:ipv6-address
+---rw dest-ipv6-address?     inet:ipv6-address
+---rw traffic-class?         uint8
+---rw flow-label?            inet:ipv6-flow-label
+---rw source-port?           inet:port-number
+---rw destination-port?      inet:port-number
+---rw protocol?              uint8
+---rw traffic-specification
+---rw interval?              uint32
+---rw max-packets-per-interval?    uint32
+---rw max-payload-size?        uint32
+---rw average-packets-per-interval? uint32
+---rw average-payload-size?      uint32
+---rw detnet-service-instance
+---rw name?                  string
+---rw flow-rank
+---rw service-rank
+---rw in-segment* [in-segment-id]
+---rw in-segment-id          uint32
+---rw (flow-type)?
+---: (IP)

```



```

+--rw (ip-flow-type)?
+--:(ipv4)
|   +--rw src-ipv4-address?    inet:ipv4-address
|   +--rw dest-ipv4-address?   inet:ipv4-address
|   +--rw dscp?                uint8
+--:(ipv6)
|   +--rw src-ipv6-address?    inet:ipv6-address
|   +--rw dest-ipv6-address?   inet:ipv6-address
|   +--rw traffic-class?       uint8
|   +--rw flow-label?          inet:ipv6-flow-label
+--rw source-port?             inet:port-number
+--rw destination-port?        inet:port-number
+--rw protocol?                uint8
+--:(MPLS)
|   +--rw service-label         uint32
+--rw service-function?        service-function-type
+--rw out-segment* [out-segment-id]
+--rw out-segment-id            uint32
+--rw detnet-service-encapsulation
|   +--rw service-label         uint32
|   +--rw control-word          uint32
+--rw detnet-transport-encapsulation
+--rw (tunnel-type)?
+--:(IPv4)
|   +--rw ipv4-encapsulation
|   |   +--rw src-ipv4-address    inet:ipv4-address
|   |   +--rw dest-ipv4-address   inet:ipv4-address
|   |   +--rw protocol            uint8
|   |   +--rw ttl?                uint8
|   |   +--rw dscp?               uint8
+--:(IPv6)
|   +--rw ipv6-encapsulation
|   |   +--rw src-ipv6-address    inet:ipv6-address
|   |   +--rw dest-ipv6-address   inet:ipv6-address
|   |   +--rw next-header         uint8
|   |   +--rw traffic-class?      uint8
|   |   +--rw flow-label?         inet:ipv6-flow-label
|   |   +--rw hop-limit?          uint8
+--:(MPLS)
|   +--rw mpls-encapsulation
|   |   +--rw label-operations* [label-oper-id]
|   |   |   +--rw label-oper-id    uint32
|   |   |   +--rw (label-actions)?
|   |   |   |   +--:(label-push)
|   |   |   |   |   +--rw label-push
|   |   |   |   |   |   +--rw label          uint32
|   |   |   |   |   |   +--rw s-bit?         boolean
|   |   |   |   |   |   +--rw tc-value?      uint8

```

```

|
|
|
|      +--rw ttl-value?      uint8
+---:(label-swap)
|      +--rw label-swap
|      +--rw out-label      uint32
|      +--rw ttl-action?    ttl-action-defi
nitition
|
|      +--rw interval?      uint32
|      +--rw max-packets-per-interval?    uint32
|      +--rw max-payload-size?    uint32
|      +--rw average-packets-per-interval?    uint32
|      +--rw average-payload-size?    uint32
+---:(end-station)
+--rw end-station
+--rw client-flow* [flow-id]
|   +--rw flow-id          uint32
|   +--rw (flow-type)?
|   |   +---:(l2-flow-identification)
|   |   |   +--rw source-mac-address?    yang:mac-address
|   |   |   +--rw destination-mac-address?    yang:mac-address
|   |   |   +--rw ethertype?            eth:ethertype
|   |   |   +--rw vlan-id?              uint16
|   |   |   +--rw pcp
|   |   +---:(l3-flow-identification)
|   |   |   +--rw (ip-flow-type)?
|   |   |   |   +---:(ipv4)
|   |   |   |   |   +--rw src-ipv4-address?    inet:ipv4-address
|   |   |   |   |   +--rw dest-ipv4-address?    inet:ipv4-address
|   |   |   |   |   +--rw dscp?                uint8
|   |   |   |   +---:(ipv6)
|   |   |   |   |   +--rw src-ipv6-address?    inet:ipv6-address
|   |   |   |   |   +--rw dest-ipv6-address?    inet:ipv6-address
|   |   |   |   |   +--rw traffic-class?        uint8
|   |   |   |   |   +--rw flow-label?          inet:ipv6-flow-label
|   |   |   +--rw source-port?            inet:port-number
|   |   |   +--rw destination-port?        inet:port-number
|   |   |   +--rw protocol?                uint8
|   +--rw traffic-specification
|   |   +--rw interval?                    uint32
|   |   +--rw max-packets-per-interval?    uint32
|   |   +--rw max-payload-size?            uint32
|   |   +--rw average-packets-per-interval?    uint32
|   |   +--rw average-payload-size?        uint32
+--rw detnet-service-instance
|   +--rw name?                          string
|   +--rw flow-rank
|   +--rw service-rank
|   +--rw in-segment* [in-segment-id]
|   |   +--rw in-segment-id              uint32
|   |   +--rw (flow-type)?

```

```

+--:(IP)
|   +--rw (ip-flow-type)?
|   |   +--:(ipv4)
|   |   |   +--rw src-ipv4-address?    inet:ipv4-address
|   |   |   +--rw dest-ipv4-address?   inet:ipv4-address
|   |   |   +--rw dscp?                 uint8
|   |   +--:(ipv6)
|   |   |   +--rw src-ipv6-address?    inet:ipv6-address
|   |   |   +--rw dest-ipv6-address?   inet:ipv6-address
|   |   |   +--rw traffic-class?       uint8
|   |   |   +--rw flow-label?          inet:ipv6-flow-label
|   +--rw source-port?                 inet:port-number
|   +--rw destination-port?            inet:port-number
|   +--rw protocol?                    uint8
+--:(MPLS)
|   +--rw service-label                uint32
+--rw service-function?                service-function-type
+--rw out-segment* [out-segment-id]
|   +--rw out-segment-id                uint32
+--rw detnet-service-encapsulation
|   +--rw service-label                uint32
|   +--rw control-word                 uint32
+--rw detnet-transport-encapsulation
|   +--rw (tunnel-type)?
|   |   +--:(IPv4)
|   |   |   +--rw ipv4-encaplustion
|   |   |   |   +--rw src-ipv4-address    inet:ipv4-address
|   |   |   |   +--rw dest-ipv4-address   inet:ipv4-address
|   |   |   |   +--rw protocol            uint8
|   |   |   |   +--rw ttl?                uint8
|   |   |   |   +--rw dscp?              uint8
|   |   +--:(IPv6)
|   |   |   +--rw ipv6-encaplustion
|   |   |   |   +--rw src-ipv6-address    inet:ipv6-address
|   |   |   |   +--rw dest-ipv6-address   inet:ipv6-address
|   |   |   |   +--rw next-header         uint8
|   |   |   |   +--rw traffic-class?     uint8
|   |   |   |   +--rw flow-label?        inet:ipv6-flow-label
|   |   |   |   +--rw hop-limit?         uint8
|   |   +--:(MPLS)
|   |   |   +--rw mpls-encaplustion
|   |   |   |   +--rw label-operations* [label-oper-id]
|   |   |   |   |   +--rw label-oper-id    uint32
|   |   |   |   +--rw (label-actions)?
|   |   |   |   |   +--:(label-push)
|   |   |   |   |   |   +--rw label-push
|   |   |   |   |   |   |   +--rw label    uint32
|   |   |   |   |   |   |   +--rw s-bit?   boolean

```

```
| | +--rw tc-value?      uint8  
| | +--rw ttl-value?    uint8  
+---:(label-swap)  
   +--rw label-swap  
       +--rw out-label     uint32  
       +--rw ttl-action?   ttl-action-definitions  
  
+--rw interval?          uint32  
+--rw max-packets-per-interval?  uint32  
+--rw max-payload-size?        uint32  
+--rw average-packets-per-interval?  uint32  
+--rw average-payload-size?         uint32
```

## 5. DetNet Configuration YANG Model

```
<CODE BEGINS> file "ietf-detnet-config@20190114.yang"
module ietf-detnet-config {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-detnet-flow-config";
  prefix "detnet-flow";

  import ietf-yang-types {
    prefix "yang";
  }

  import ietf-inet-types{
    prefix "inet";
  }

  import ietf-ethertypes {
    prefix "eth";
  }

  organization "IETF DetNet Working Group";

  contact
    "WG Web:  <http://tools.ietf.org/wg/detnet/>
    WG List:  <mailto: detnet@ietf.org>
    WG Chair: Lou Berger
               <mailto:lberger@labn.net>

               Janos Farkas
               <janos.farkas@ericsson.com>

    Editor:   Xuesong Geng
               <mailto:gengxuesong@huawei.com>

    Editor:   Mach Chen
               <mailto:mach.chen@huawei.com>
```

```
Editor:   Zhenqiang Li
         <lizhenqiang@chinamobile.com>

Editor:   Reshad Rahman
         <rrahman@cisco.com>";
description
  "This YANG module describes the parameters needed
  for DetNet flow configuration and flow status
  reporting.";

revision "2018-09-10" {
  description "initial revision";
  reference "RFC XXXX: draft-geng-detnet-config-yang-05";
}

identity detnet-node-role {
  description
    "base detnet-node-role";
}

identity end-station {
  base detnet-node-role;
  description
    "Commonly called a 'host' in IETF documents,
    and an 'end station' is IEEE 802 documents.
    End systems of interest to this document
    are either sources or destinations of DetNet
    flows. And end system may or may not be
    DetNet transport layer aware or DetNet
    service layer aware.";
}

identity edge-node {
  base detnet-node-role;
  description
    "An instance of a DetNet relay node that
    includes either a DetNet service layer proxy
    function for DetNet service protection (e.g.
    the addition or removal of packet sequencing
    information) for one or more end systems, or
    starts or terminate congestion protection at
    the DetNet transport layer, analogous to a
    Label Edge Router (LER).";
}

identity relay-node {
  base detnet-node-role;
  description
```

```
"A DetNet node including a service layer
function that interconnects different DetNet
transport layer paths to provide service
protection. A DetNet relay node can be a bridge,
a router, a firewall, or any other system that
participates in the DetNet service layer. It
typically incorporates DetNet transport layer
functions as well, in which case it is
collocated with a transit node.";
}

identity transit-node {
  base detnet-node-role;
  description
    "A node operating at the DetNet transport layer,
    that utilizes link layer and/or network layer
    switching across multiple links and/or
    sub-networks to provide paths for DetNet
    service layer functions. Optionally provides
    congestion protection over those paths. An MPLS
    LSR is an example of a DetNet transit node.";
}

identity ttl-action {
  description
    "Base identity from which all TTL
    actions are derived.";
}

identity no-action {
  base "ttl-action";
  description
    "Do nothing regarding the TTL.";
}

identity copy-to-inner {
  base "ttl-action";
  description
    "Copy the TTL of the outer header
    to the inner header.";
}

identity decrease-and-copy-to-inner {
  base "ttl-action";
  description
    "Decrease TTL by one and copy the TTL
    to the inner header.";
}
```

```
typedef ttl-action-definition {
    type identityref {
        base "ttl-action";
    }
    description
        "TTL action definition.";
}

identity detnet-transport-layer {
    description
        "The layer that optionally provides congestion
        protection for DetNet flows over paths provided
        by the underlying network.";
}

identity detnet-service-layer {
    description
        "The layer at which service protection is
        provided, either packet sequencing, replication,
        and elimination or packet encoding";
}

typedef service-function-type {
    type enumeration {
        enum replication {
            description
                "A Packet Replication Function (PRF) replicates
                DetNet flow packets and forwards them to one or
                more next hops in the DetNet domain. The number
                of packet copies sent to each next hop is a
                DetNet flow specific parameter at the node doing
                the replication. PRF can be implemented by an
                edge node, a relay node, or an end system";
        }
        enum elimination {
            description
                "A Packet Elimination Function (PEF) eliminates
                duplicate copies of packets to prevent excess
                packets flooding the network or duplicate
                packets being sent out of the DetNet domain.
                PEF can be implemented by an edge node, a relay
                node, or an end system.";
        }
        enum ordering {
            description
                "A Packet Ordering Function (POF) re-orders
                packets within a DetNet flow that are received
                out of order. This function can be implemented
```

```
        by an edge node, a relay node, or an end system.";
    }
    enum elimination-ordering {
        description
            "A combination of PEF and POF that can be
             implemented by an edge node, a relay node, or
             an end system.";
    }
    enum elimination-replication {
        description
            "A combination of PEF and PRF that can be
             implemented by an edge node, a relay node, or
             an end system";
    }
    enum elimination-ordering-replicaiton {
        description
            "A combination of PEF, POF and PRF that can be
             implemented by an edge node, a relay node, or
             an end system";
    }
}
description
    "DetNet service function and function combination
     types.";
}

grouping detnet-transport-qos {
    description
        "DetNet transport tunnel QoS attributes.";
    uses traffic-specification;
}

grouping ipv4-header {
    description
        "The IPv4 header encapsulation information.";
    leaf src-ipv4-address {
        type inet:ipv4-address;
        mandatory true;
        description
            "The source IP address of the header.";
    }
    leaf dest-ipv4-address {
        type inet:ipv4-address;
        mandatory true;
        description
            "The destination IP address of the header.";
    }
    leaf protocol {
```



```
        type uint8;
        mandatory true;
        description
            "The protocol id of the header.";
    }
    leaf ttl {
        type uint8;
        description
            "The TTL of the header.";
    }
    leaf dscp {
        type uint8;
        description
            "The DSCP field of the header.";
    }
}

grouping ipv6-header {
    description
        "The IPv6 header encapsulation information.";
    leaf src-ipv6-address {
        type inet:ipv6-address;
        mandatory true;
        description
            "The source IP address of the header.";
    }
    leaf dest-ipv6-address {
        type inet:ipv6-address;
        mandatory true;
        description
            "The destination IP address of the header.";
    }
    leaf next-header {
        type uint8;
        mandatory true;
        description
            "The next header of the IPv6 header.";
    }
    leaf traffic-class {
        type uint8;
        description
            "The traffic class value of the header.";
    }
    leaf flow-label {
        type inet:ipv6-flow-label;
        description
            "The flow label of the header.";
    }
}
```

```
    leaf hop-limit {
      type uint8 {
        range "1..255";
      }
      description
        "The hop limit of the header.";
    }
  }

  grouping mpls-header {
    description
      "The MPLS encapsulation header information.";
    list label-operations {
      key "label-oper-id";
      description
        "Label operations.";
      leaf label-oper-id {
        type uint32;
        description
          "An optional identifier that points
            to a label operation.";
      }
      choice label-actions {
        description
          "Label action options.";
        case label-push {
          container label-push {
            description
              "Label push operation.";
            leaf label {
              type uint32;
              mandatory true;
              description
                "The label to be pushed.";
            }
            leaf s-bit {
              type boolean;
              description
                "The s-bit of the label to be pushed.";
            }
          }
          leaf tc-value {
            type uint8;
            description
              "The traffic class value of the label
                to be pushed.";
          }
          leaf ttl-value {
            type uint8;
          }
        }
      }
    }
  }
}
```

```

        description
            "The TTL value of the label to be
            pushed.";
    }
}
}
case label-swap {
    container label-swap {
        description
            "Label swap operation.";
        leaf out-label {
            type uint32;
            mandatory true;
            description
                "The out MPLS label.";
        }
        leaf ttl-action {
            type ttl-action-definition;
            description
                "The label ttl actions:
                - No-action, or
                - Copy to inner label, or
                - Decrease (the in label) by 1 and
                copy to the out label.";
        }
    }
}
}
}
}
}

grouping mpls-detnet-header {
    description
        "The MPLS DetNet encapsulation header information.";
    leaf service-label {
        type uint32;
        mandatory true;
        description
            "The service label.";
    }
    leaf control-word {
        type uint32;
        mandatory true;
        description
            "The control word of the DetNet header.";
    }
}
}

```

```
grouping transport-tunnel-encap{
  description
    "Defines the transport tunnel encapsulation
    header.";
  choice tunnel-type {
    description
      "Tunnel type includes: IPv4, IPv6, MPLS.";
    case IPv4 {
      description
        "IPv4 tunnel.";
      container ipv4-encapsulation {
        description
          "IPv4 encapsulation.";
        uses ipv4-header;
      }
    }
    case IPv6 {
      description
        "IPv6 tunnel.";
      container ipv6-encapsulation {
        description
          "IPv6 encapsulation.";
        uses ipv6-header;
      }
    }
    case MPLS {
      description
        "MPLS tunnel.";
      container mpls-encapsulation {
        description
          "MPLS encapsulation.";
        uses mpls-header;
      }
    }
  }
}

grouping detnet-transport-instance {
  description
    "An instance of the DetNet transport layer, which
    depends on the specific data plane that is used
    as the underlay tunnel.";
  uses transport-tunnel-encap;
  uses detnet-transport-qos;
}

grouping ip-flow-identification {
  description
```

```
    "IP flow identification.";
choice ip-flow-type {
  description
    "IP flow types: IPv4, IPv6.";
  case ipv4 {
    description
      "IPv4 flow identification.";
    leaf src-ipv4-address {
      type inet:ipv4-address;
      description
        "The source IP address of the header.";
    }
    leaf dest-ipv4-address {
      type inet:ipv4-address;
      description
        "The destination IP address of the header.";
    }
    leaf dscp {
      type uint8;
      description
        "The DSCP field of the header.";
    }
  }
  case ipv6 {
    description
      "IPv6 flow identification.";
    leaf src-ipv6-address {
      type inet:ipv6-address;
      description
        "The source IP address of the header.";
    }
    leaf dest-ipv6-address {
      type inet:ipv6-address;
      description
        "The destination IP address of the header.";
    }
    leaf traffic-class {
      type uint8;
      description
        "The traffic class value of the header.";
    }
    leaf flow-label {
      type inet:ipv6-flow-label;
      description
        "The flow label of the header.";
    }
  }
}
```

```
    leaf source-port {
      type inet:port-number;
      description
        "The source port number.";
    }
    leaf destination-port {
      type inet:port-number;
      description
        "The destination port number.";
    }
    leaf protocol {
      type uint8;
      description
        "The protocol id of the header.";
    }
  }
}

grouping l3-flow-identification {
  description
    "Layer 3 flow identification in the DetNet
    domain.";
  choice flow-type {
    description
      "L3 DetNet flow types: IP and MPLS.";
    case IP {
      description
        "IP (IPv4 or IPv6) flow identification.";
      uses ip-flow-identification;
    }
    case MPLS {
      description
        "MPLS flow identification.";
      leaf service-label {
        type uint32;
        mandatory true;
        description
          "The service label.";
      }
    }
  }
}
} //l3-flow-identification

grouping in-segments {
  description
    "From a receiving node point of view, In-segments
    are a set of instances of a DetNet flow at the
    receiving node. This occurs when Packet Replication
    Function (PRF) is enabled at an upstream node or
```

```
        multiple flows map/aggregate to a single DetNet
        flow.";
    list in-segment {
        key "in-segment-id";

        description
            "A list of in segments, there will be
            multiple in-segments for a DetNet flow
            when PRF and PEF enabled.";

        leaf in-segment-id {
            type uint32;
            description
                "in-segment identifier.";
        }

        uses l3-flow-identification;

        leaf service-function {
            type service-function-type;
            description
                "DetNet service function indication.";
        }
    }
}

grouping out-segments {
    description
        "Out-segments are a set of instances of
        a DetNet flow, this occurs when implement
        packet replication function, where an
        in-segment of a DetNet flow is replicated
        to multiple out-segments.";

    list out-segment {
        key "out-segment-id";
        description
            "A list of segments, there will be multiple
            out-segments when perform PRF.";
        leaf out-segment-id {
            type uint32;
            description
                "The out-segment identifier";
        }

        container detnet-service-encapsulation {
            description
                "Only MPLS based DetNet defines DetNet
```

```

        service layer. The service encapsulation
        includes service label and control word.";
    uses mpls-detnet-header;
}

container detnet-transport-encapsulation {
    description
        "Each out-segment corresponds to a
        transport instance.";
    uses detnet-transport-instance;
}
}

grouping detnet-service-instance{
    description
        "An end-2-end DetNet service is consisted of
        multiple segments. The concept of segment is
        similar to PW segment. For DetNet, since the
        existing of PREOF, there could be three cases:
        1 - One in-segment maps to multiple
            out-segments, when implement PRF;
        2 - Multiple in-segments map to one
            out-segment, when implement PEF;
        3 - Multiple in-segments map to multiple
            out-segments, when implement a combination
            of PEF and PRF.";

    leaf name {
        type string;
        description
            "The name of the service instance. This MUST
            be unique across all service instances in
            a given network device.";
    }
    container flow-rank{
        description
            "TBD based on the data plane solution.";
    }
    container service-rank{
        description
            "TBD based on the data plane solution.";
    }
    uses in-segments;
    uses out-segments;
}

grouping l2-flow-identification-at-uni {

```



```
description
  "Layer 2 flow identification at UNI.";
leaf source-mac-address {
  type yang:mac-address;
  description
    "The source MAC address used for
    flow identification.";
}
leaf destination-mac-address {
  type yang:mac-address;
  description
    "The destination MAC address used for
    flow identification.";
}

leaf ethertype {
  type eth:ethertype;
  description
    "The Ethernet Type (or Length) value represented
    in the canonical order defined by IEEE 802.
    The canonical representation uses lowercase
    characters.";
  reference
    "IEEE 802-2014 Clause 9.2";
}

leaf vlan-id {
  type uint16 {
    range "1..4094";
  }
  description
    "Vlan Identifier used for L2 flow identification.";
}
container pcg {
  //Todo
  description
    "PCP used for L2 flow identification.";
}
}

grouping l3-flow-identification-at-uni {
  description
    "Layer 3 flow identification at UNI.";
  uses ip-flow-identification;
}

grouping traffic-specification {
  description
```

```
    "traffic-specification specifies how the Source
      transmits packets for the flow. This is the
      promise/request of the Source to the network.
      The network uses this traffic specification
      to allocate resources and adjust queue
      parameters in network nodes.";
  reference
    "draft-ietf-detnet-flow-information-model";

  leaf interval {
    type uint32;
    description
      "The period of time in which the traffic
        specification cannot be exceeded";
  }
  leaf max-packets-per-interval{
    type uint32;
    description
      "The maximum number of packets that the
        source will transmit in one Interval.";
  }
  leaf max-payload-size{
    type uint32;
    description
      "The maximum payload size that the source
        will transmit.";
  }
  leaf average-packets-per-interval {
    type uint32;
    description
      "The average number of packets that the
        source will transmit in one Interval";
  }
  leaf average-payload-size {
    type uint32;
    description
      "The average payload size that the
        source will transmit.";
  }
}

grouping client-flows-at-uni {
  description
    "The attributes of the client flow at UNI. When
      flow aggregation is enabled at ingress, multiple
      client flows map to a DetNet service instance.";
  list client-flow {
    key "flow-id";
```

```
description
  "A list of client flows.";
leaf flow-id {
  type uint32;
  description
    "Flow identifier that is unique in a network
    device for client flow identification";
}
choice flow-type{
  description
    "Client flow type: layer 2 flow, layer 3
    flow.";
  case l2-flow-identification {
    description
      "Ethernet flow identification.";
    uses l2-flow-identification-at-uni;
  }
  case l3-flow-identification {
    description
      "layer 3 flow identification, including
      IPv4, IPv6 and MPLS.";
    uses l3-flow-identification-at-uni;
  }
}
container traffic-specification {
  description
    "The traffic specification of the client flow.";
  uses traffic-specification;
}
}

grouping detnet-service-proxy-instance {
  description
    "Maps between App-flows and DetNet flows";
  uses client-flows-at-uni;
  container detnet-service-instance {
    description
      "A DetNet service instance.";
    uses detnet-service-instance;
  }
}

container detnet-flow{
  description
    "DetNet flow configuration and status reporting.";
  choice detnet-node-role{
    description
```

```

    "Depends on the role of a node to configure
    corresponding flow parameters.";

case transit-node{
  description
    "DetNet flow configuration parameters for
    transit nodes.";
  container transit-node {
    description
      "transit node container.";
    uses detnet-transport-qos;
  }
}
case relay-node{
  description
    "DetNet flow configuration parameters for
    relay nodes.";
  container relay-node {
    description
      "Relay node container.";
    uses detnet-service-instance;
  }
}
case edge-node{
  description
    "DetNet flow configuration parameters for
    edge nodes.";
  container edge-node {
    description
      "Edge node container.";
    uses detnet-service-proxy-instance;
  }
}
case end-station {
  description
    "DetNet flow configuration parameters for
    end stations.";
  container end-station {
    description
      "End station container.";
    uses detnet-service-proxy-instance;
  }
}
}
}
}
<CODE ENDS>

```

## 6. DetNet Configuration Model Classification

This section defines three classes of DetNet configuration model: fully distributed configuration model, fully centralized configuration model, hybrid configuration model, based on different network architectures, showing how configuration information exchanges between various entities in the network.

### 6.1. Fully Distributed Configuration Model

In a fully distributed configuration model, UNI information is transmitted over DetNet UNI protocol from the user side to the network side; then UNI information and network configuration information propagate in the network over distributed control plane protocol. For example:

- 1) IGP collects topology information and DetNet capabilities of network([I-D.geng-detnet-info-distribution]);
- 2) Control Plane of the Edge Node(Ingress) receives a flow establishment request from UNI and calculates a/some valid path(s);
- 3) Using RSVP-TE, Edge Node(Ingress) sends a PATH message with explicit route. After receiving the PATH message, the other Edge Node(Egress) sends a Resv message with distributed label and resource reservation request.

Current distributed control plane protocol, e.g., RSVP-TE[RFC3209], SRP[IEEE802.1Qcc], can only reserve bandwidth along the path, while the configuration of a fine-grained schedule, e.g., Time Aware Shaping(TAS) defined in [IEEE802.1Qbv], is not supported.

The fully distributed configuration model is not covered by this draft. It should be discussed in the future DetNet control plane work.

### 6.2. Fully Centralized Configuration Model

In the fully centralized configuration model, UNI information is transmitted from Centralized User Configuration (CUC) to Centralized Network Configuration(CNC). Configurations of routers for DetNet flows are performed by CNC with network management protocol. For example:

- 1) CNC collects topology information and DetNet capability of network through Netconf;

2) CNC receives a flow establishment request from UNI and calculates a/some valid path(s);

3) CNC configures the devices along the path for flow transmission.

### 6.3. Hybrid Configuration Model

In the hybrid configuration model, controller and control plane protocols work together to offer DetNet service, and there are a lot of possible combinations. For example:

1) CNC collects topology information and DetNet capability of network through IGP/BGP-LS;

2) CNC receives a flow establishment request from UNI and calculates a/some valid path(s);

3) Based on the calculation result, CNC distributes flow path information to Edge Node(Ingress) and other information(e.g. replication/elimination) to the relevant nodes.

4) Using RSVP-TE, Edge Node(Ingress) sends a PATH message with explicit route. After receiving the PATH message, the other Edge Node(Egress) sends a Resv message with distributed label and resource reservation request.

or

1) Controller collects topology information and DetNet capability of network through IGP/BGP-LS;

2) Control Plane of Edge Node(Ingress) receives a flow establishment request from UNI;

3) Edge Node(Ingress) sends the path establishment request to CNC through PCEP;

4) After Calculation, CNC sends back the path information of the flow to the Edge Node(Ingress) through PCEP;

5) Using RSVP-TE, Edge Node(Ingress) sends a PATH message with explicit route. After receiving the PATH message, the other Edge Node(Egress) sends a Resv message with distributed label and resource reservation request.

There are also other variations that can be included in the hybrid model. This draft can not cover all the control plane data needed

in hybrid configuration models. Every solution has there own mechanism and corresponding parameters to make it work.

Editor's Note:

1. There are a lot of optional DetNet configuration models, and different scenario in different use case can choose one of them based on its conditions. Maybe next step of the work is to pick up one or more typical scenarios and give a practical solution.

2. [IEEE802.1Qcc] also defines three TSN configuration models: fully-centralized model, fully-distributed model, centralized Network / distributed User Model. This section defines the configuration model roughly the same, to keep the design of L2 and L3 in the same structure. Hybrid configuration model is slightly different from the 'centralized Network / distributed User Model'. The hybrid configuration model intends to contain more variations.

## 7. Open Issues

There are some open issues that are still under discussion:

- o The Relationship with 802.1 TSN YANG models is TBD. TSN YANG models include: P802.1Qcw, which defines TSN YANG for Qbv, Qbu, and Qci, and P802.1CBcv, which defines YANG for 802.1CB. The possible problem here is how to avoid possible overlap among yang models defined in IETF and IEEE. A common YANG model may be defined in the future to shared by both TSN and DetNet. More discussion are needed here.
- o How to support DetNet OAM is TBD.

These issues will be resolved in the following versions of the draft.

## 8. IANA Considerations

This document makes no request of IANA.

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

## 9. Security Considerations

<TBD>

## 10. Acknowledgements

## 11. References

## 11.1. Normative References

- [I-D.finn-detnet-bounded-latency]  
Finn, N., Boudec, J., Mohammadpour, E., Zhang, J., Varga, B., and J. Farkas, "DetNet Bounded Latency", draft-finn-detnet-bounded-latency-02 (work in progress), October 2018.
- [I-D.ietf-detnet-architecture]  
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-10 (work in progress), December 2018.
- [I-D.ietf-detnet-dp-sol-ip]  
Korhonen, J. and B. Varga, "DetNet IP Data Plane Encapsulation", draft-ietf-detnet-dp-sol-ip-01 (work in progress), October 2018.
- [I-D.ietf-detnet-dp-sol-mpls]  
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane Encapsulation", draft-ietf-detnet-dp-sol-mpls-01 (work in progress), October 2018.
- [I-D.ietf-detnet-flow-information-model]  
Farkas, J., Varga, B., Cummings, R., Jiang, Y., and Y. Zha, "DetNet Flow Information Model", draft-ietf-detnet-flow-information-model-02 (work in progress), October 2018.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.



## 11.2. Informative References

- [I-D.geng-detnet-info-distribution]  
Geng, X., Chen, M., and Z. Li, "IGP-TE Extensions for DetNet Information Distribution", draft-geng-detnet-info-distribution-03 (work in progress), October 2018.
- [I-D.ietf-detnet-use-cases]  
Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-20 (work in progress), December 2018.
- [I-D.ietf-teas-yang-te]  
Saad, T., Gandhi, R., Liu, X., Beeram, V., Shah, H., and I. Bryskin, "A YANG Data Model for Traffic Engineering Tunnels and Interfaces", draft-ietf-teas-yang-te-17 (work in progress), October 2018.
- [I-D.ietf-teas-yang-te-topo]  
Liu, X., Bryskin, I., Beeram, V., Saad, T., Shah, H., and O. Dios, "YANG Data Model for Traffic Engineering (TE) Topologies", draft-ietf-teas-yang-te-topo-18 (work in progress), June 2018.
- [I-D.thubert-tsvwg-detnet-transport]  
Thubert, P., "A Transport Layer for Deterministic Networks", draft-thubert-tsvwg-detnet-transport-01 (work in progress), October 2017.
- [I-D.varga-detnet-service-model]  
Varga, B. and J. Farkas, "DetNet Service Model", draft-varga-detnet-service-model-02 (work in progress), May 2017.
- [IEEE802.1CB]  
"IEEE, "Frame Replication and Elimination for Reliability (IEEE Draft P802.1CB)", 2017,  
<<http://www.ieee802.org/1/files/private/cb-drafts/>>.", 2016.
- [IEEE802.1Q-2014]  
"IEEE, "IEEE Std 802.1Q Bridges and Bridged Networks", 2014, <<http://ieeexplore.ieee.org/document/6991462/>>.", 2014.

- [IEEE802.1Qbu]  
"IEEE, "IEEE Std 802.1Qbu Bridges and Bridged Networks - Amendment 26: Frame Preemption", 2016,  
<<http://ieeexplore.ieee.org/document/7553415/>>.", 2016.
- [IEEE802.1Qbv]  
"IEEE, "IEEE Std 802.1Qbu Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic", 2015,  
<<http://ieeexplore.ieee.org/document/7572858/>>.", 2016.
- [IEEE802.1Qcc]  
"IEEE, "Stream Reservation Protocol (SRP) Enhancements and Performance Improvements (IEEE Draft P802.1Qcc)", 2017,  
<<http://www.ieee802.org/1/files/private/cc-drafts/>>.".
- [IEEE802.1Qch]  
"IEEE, "Cyclic Queuing and Forwarding (IEEE Draft P802.1Qch)", 2017,  
<<http://www.ieee802.org/1/files/private/ch-drafts/>>.", 2016.
- [IEEE802.1Qci]  
"IEEE, "Per-Stream Filtering and Policing (IEEE Draft P802.1Qci)", 2016,  
<<http://www.ieee802.org/1/files/private/ci-drafts/>>.", 2016.
- [RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <<https://www.rfc-editor.org/info/rfc3209>>.
- [RFC4875] Aggarwal, R., Ed., Papadimitriou, D., Ed., and S. Yasukawa, Ed., "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, DOI 10.17487/RFC4875, May 2007, <<https://www.rfc-editor.org/info/rfc4875>>.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <<https://www.rfc-editor.org/info/rfc8342>>.

Authors' Addresses

Xuesong Geng  
Huawei Technologies

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen  
Huawei Technologies

Email: mach.chen@huawei.com

Zhenqiang Li  
China Mobile

Email: lizhenqiang@chinamobile.com

Reshad Rahman  
Cisco Systems

Email: rrahman@cisco.com

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: September 9, 2019

L. Qiang, Ed.  
X. Geng  
B. Liu  
T. Eckert, Ed.  
Huawei  
L. Geng  
China Mobile  
March 8, 2019

Large-Scale Deterministic IP Network  
draft-qiang-detnet-large-scale-detnet-04

Abstract

This document presents the overall framework and key method for Large-scale Deterministic Network (LDN). LDN can provide bounded latency and delay variation (jitter) without requiring precise time synchronization among nodes, or per-flow state in transit 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 September 9, 2019.

Copyright Notice

Copyright (c) 2019 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
1.2. Terminology & Abbreviations . . . . .	3
2. Overview . . . . .	3
2.1. Summary . . . . .	4
2.2. Background . . . . .	4
2.2.1. Deterministic End-to-End Latency . . . . .	4
2.2.2. Hop-by-Hop Delay . . . . .	4
2.2.3. Cyclic Forwarding . . . . .	5
2.2.4. Co-Existence with Non-Deterministic Traffic . . . . .	6
2.3. System Components . . . . .	6
3. LDN Forwarding Mechanism . . . . .	7
3.1. Cyclic Queues . . . . .	8
3.2. Cycle Mapping . . . . .	9
4. Performance Analysis . . . . .	11
4.1. Queueing Delay . . . . .	11
4.2. Jitter . . . . .	11
5. IANA Considerations . . . . .	13
6. Security Considerations . . . . .	13
7. Acknowledgements . . . . .	13
8. Normative References . . . . .	14
Authors' Addresses . . . . .	14

## 1. Introduction

This document explores the DetNet forwarding over large-scale network. In contrast to TSN that deployed in LAN, DetNet is expected to be deployed in larger scale network that has the following features:

- o a large number of network devices
- o the distance between two network devices is long
- o a lot of deterministic flows on the network

These above features will bring the following challenges to DetNet forwarding:

- o difficult to achieve precise time synchronization among all nodes
- o long link propagation delay may introduce bigger jitter

- o per-flow state is un-scalable

Motivated by these challenges, this document presents a Large-scale Deterministic Network (LDN) mechanism. As [draft-ietf-detnet-problem-statement] indicates, deterministic forwarding can only apply on flows with well-defined traffic characteristics. The traffic characteristics of DetNet flow has been discussed in [draft-ietf-detnet-architecture], that could be achieved through shaping at Ingress node or up-front commitment by application. LDN assumes that DetNet flows follow some specific traffic patterns accordingly.

### 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.

### 1.2. Terminology & Abbreviations

This document uses the terminology defined in [draft-ietf-detnet-architecture].

TSN: Time Sensitive Network

PQ: Priority Queuing

CQF: Cyclic Queuing and Forwarding

LDN: Large-scale Deterministic Network

DSCP: Differentiated Services Code Point

EXP: Experimental

TC: Traffic Class

T: the length of a cycle

H: the number of hops

## 2. Overview

## 2.1. Summary

In LDN, nodes (network devices) have synchronized frequency, and each node forwards packets in a slotted fashion based on a cycle identifiers carried in packets. Ingress nodes or senders have a function called gate to shape/condition traffic flows. Except for this gate function, the LDN has no awareness of individual flows.

## 2.2. Background

This section motivates the design choices taken by the proposed solution and gives the necessary background for deterministic delay based forwarding plane designs.

### 2.2.1. Deterministic End-to-End Latency

Bounded delay is delay that has a deterministic upper and lower bound.

The delay for packets that need to be forwarded with deterministic delay needs to be deterministic on every hop. If any hop in the network introduces non-deterministic delay, then the network itself can not deliver a deterministic delay service anymore.

### 2.2.2. Hop-by-Hop Delay

Consider a simple example shown in Figure 1, where Node X has 10 receiving interfaces and one outgoing interface I all of the same speed. There are 10 deterministic traffic flows, each consuming 5% of a link's bandwidth, one from each receiving interface to the outgoing interface.

Node X sends 'only' 50% deterministic traffic to interface I, so there is no ongoing congestion, but there is added delay. If the arrival time of packets for these 10 flows into X is uncontrolled, then the worst case is for them to all arrive at the same time. One packet has to wait in X until the other 9 packets are sent out on I, resulting in a worst case deterministic delay of 9 packets serialization time. On the next hop node Y downstream from X, this problem can become worse. Assume Y has 10 upstream nodes like X, the worst case simultaneous burst of packets is now 100 packets, or a 99 packet serialization delay as the worst case upper bounded delay incurred on this hop.

To avoid the problem of high upper bound end-to-end delay, traffic needs to be conditioned/interleaved on every hop. This allows to create solutions where the per-hop-delay is bounded purely by the

The diagram illustrates a 4-bit bus system connecting two nodes, Node X and Node Y, through a central interface. The bus is represented by a central horizontal line with arrows pointing to the nodes. Node X has two outputs, A1 and A0, and two inputs, B1 and B0. Node Y has two outputs, C1 and C0, and two inputs, D1 and D0. The bus is labeled "Interface I" in the center.



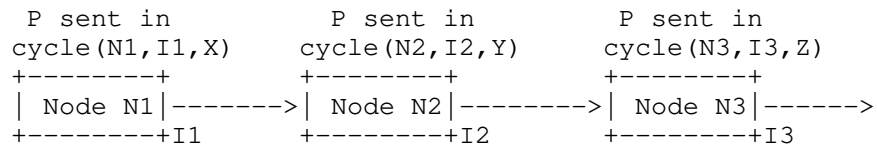


Figure 2: Cyclic Forwarding

#### 2.2.4. Co-Existence with Non-Deterministic Traffic

Traffic with deterministic delay requirements can co-exist with traffic only requiring non-deterministic delay by using packet scheduling where the delay incurred by non-deterministic packets is deterministic for the deterministic traffic (and low). If LDN is deployed together with such non-deterministic delay traffic then such a scheme must be supported by the forwarding plane. A simple approach for the delay incurred on the sending interface of a deterministic node due to non-deterministic traffic is to serve deterministic traffic via a strict, highest-priority queue and include the worst case delay of a currently serialized non-deterministic packet into the deterministic delay budget of the node. Similar considerations apply to the internal processing delays in a node.

#### 2.3. System Components

The Figure 3 shows an overview of the components considered in this document system and how they interact.

A network topology of nodes, Ingress, Core and Egress support a method for cyclic forwarding to enable LDN. This forwarding requires no per-flow state on the nodes, and tolerates loss time synchronization.

Ingress edge nodes may support the (G)ate function to shape traffic from sources into the desired traffic characteristics, unless the source itself has such function. Per-flow state is required on the ingress edge node. LDN should work with some resource reservation methods, that will be not discussed in this document.

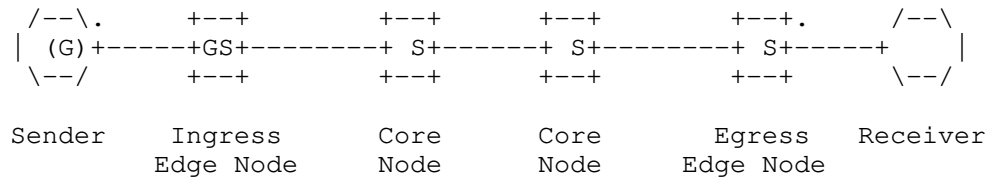
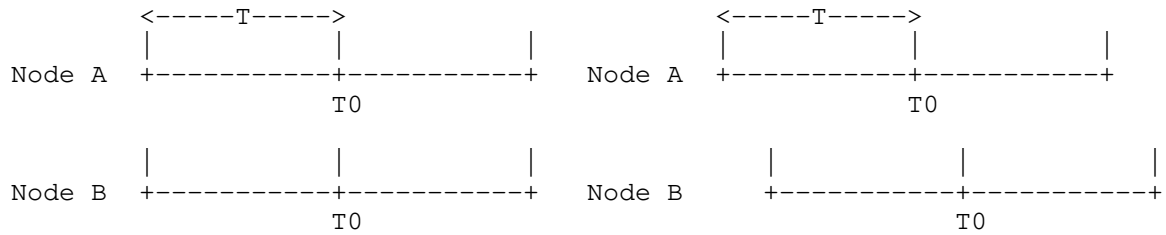


Figure 3: Overview of LDN

### 3. LDN Forwarding Mechanism

DetNet aims at providing deterministic service over large scale network. In such large scale network, it is difficulty to get precise time synchronization among numerous devices. To reduce requirements, the forwarding mechanism described in this document assumes only frequency synchronization but not time synchronization across nodes: nodes maintain the same clock frequency  $1/T$ , but do not require the same time as shown in Figure 4.



T: length of a cycle  
 T0: timestamp

Figure 4: Time Synchronization &amp; Frequency Synchronization

IEEE 802.1 CQF is an efficient forwarding mechanism in TSN that guarantees bounded end-to-end latency. CQF is designed for limited scale networks. Time synchronization is required, and the link propagation delay is required to be smaller than a cycle length  $T$ . Considering the large scale network deployment, the proposed LDN Forwarding mechanism permits frequency synchronization and link propagation delay may exceed  $T$ . Besides these two points, CQF and the asynchronous forwarding of LDN are very similar.

Figure 5 compares CQF and LDN through an example. Suppose Node A is the upstream node of Node B. In CQF, packets sent from Node A at

cycle  $x$ , will be received by Node B at the same cycle, then further be sent to downstream node by Node B at cycle  $x+1$ .

In LDN, due to long link propagation delay and frequency synchronization, Node B will receive packets from Node A at different cycle denoted by  $y$ , then re-send out at cycle  $y+1$ . The cycle mapping relationship (e.g.,  $x \rightarrow y+1$ ) exists between any pair of neighbor nodes. With this kind of cycle mapping, the receiving node can easily figure out when the received packets should be sent out, the only requirement is to carry the cycle identifier of sending node in the packets.

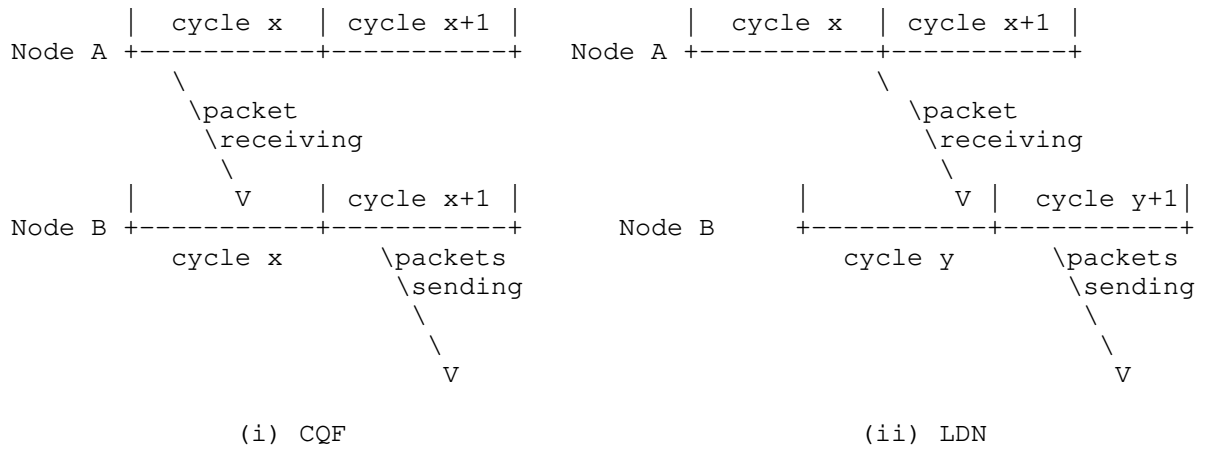


Figure 5: CQF & LDN

### 3.1. Cyclic Queues

In CQF each port needs to maintain 2 (or 3) queues, one receiving queue is used to buffer newly received packets, one sending queue is used to store the packets that are going to be sent out, one more queue may be needed to avoid output starvation [scheduled-queues].

In LDN, at least 3 cyclic queues (2 receiving queues and 1 sending queue) are maintained for each port on a node. A cyclic queue corresponds to a cycle. As Figure 6 illustrated, the downstream Node B may receive packets sent at two different cycles from Node A due to the absence of time synchronization. Following the cycle mapping (i.e.,  $x \rightarrow y+1$ ), packets that carry cycle identifier  $x$  should be sent out by Node B at cycle  $y+1$ , and packets that carry cycle identifier  $x+1$  should be sent out by Node B at cycle  $y+2$ . Therefore, 2 receiving queues are needed to store the received packets, one is for the packets that carry cycle identifier  $x$ , another one is for the

packets that carry cycle identifier  $x+1$ . Plus one sending queue, each port needs at least 3 cyclic queues in LDN. In order to absorb more link delay variation (such as on radio interface), more queues may be necessary.

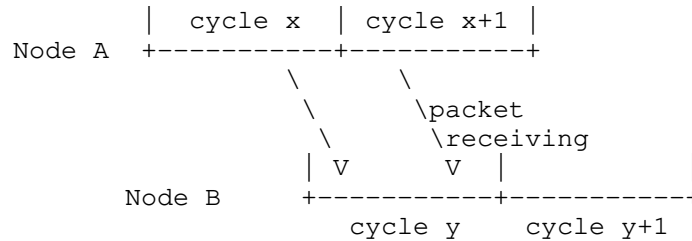


Figure 6: An example illustrates for 2 receiving queue in LDN

### 3.2. Cycle Mapping

The cycle mapping relationship (e.g.,  $x \rightarrow y+1$ ) exists between any pair of neighbor nodes, that could be configured through control plane or self-studied in data plane. As Figure 7 shows, the cycle mapping relationship instructs the packet forwarding in two modes -- swap mode or stack mode.

- o In swap mode, node stores the cycle mapping relationship locally. After receiving a packet carrying a cycle identifier, the node will check its cycle mapping relationship table, swap the cycle identifier with a new cycle identifier, then put the packet into an appropriate queue. A path with dedicated resource needs to be established first, then packet is forwarded along the path in swap mode.
- o In stack mode, a central controller computes the cycle identifier of every node, which ensures that there is no flow confliction along the path and satisfies the end-to-end latency requirement. The cycle identifiers are encapsulated into the packet in the ingress. No other status information needs to be maintained in the intermediate nodes.

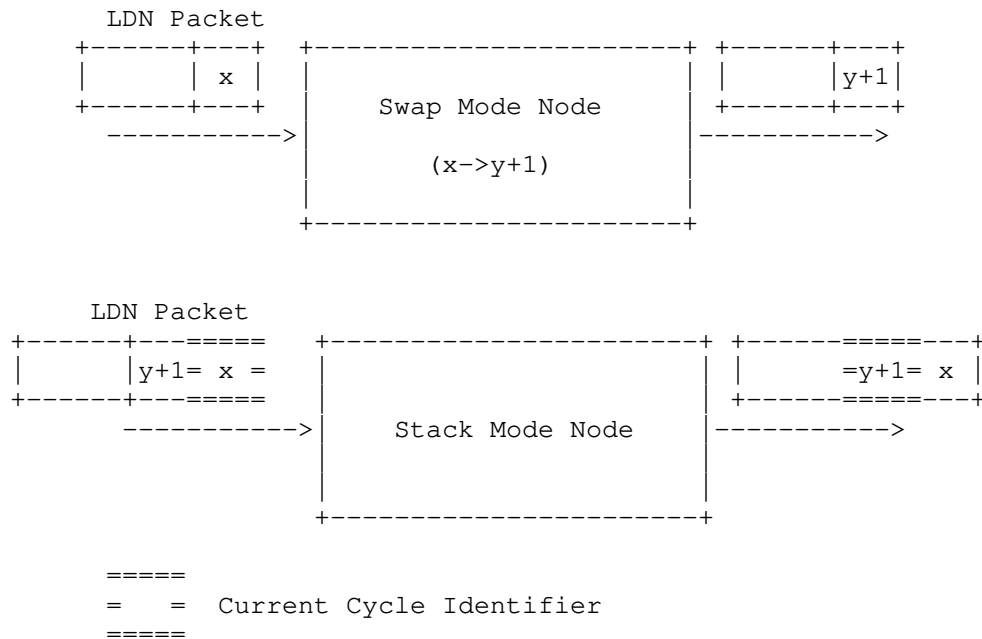


Figure 7: Two Modes

As section 3.1 illustrates, there are 3 (or 4) different queues at each port. Therefore, the cycle identifier should be able to express 3 (or 4) different values, each value corresponds to a queue. That means minimal 2 bits are needed to identify different cycles between a pair of neighboring nodes. This document does not yet aim to propose one, but gives an (incomplete) list of ideas:

- o DSCP of IPv4 Header
- o Traffic Class of IPv6 Header
- o TC of MPLS Header (used to be EXP)
- o IPv6 Extension Header
- o UDP Option
- o SID of SRv6
- o Reserved of SRH
- o TLV of SRv6

- o TC of SR-MPLS Header (used to be EXP)
- o 3 (or 4) labels/adjacency SIDs for SR-MPLS

#### 4. Performance Analysis

##### 4.1. Queueing Delay

Figure 8 describes one-hop packet forwarding delay, that mainly consisted of A->B link propagation delay and queueing delay in Node B.

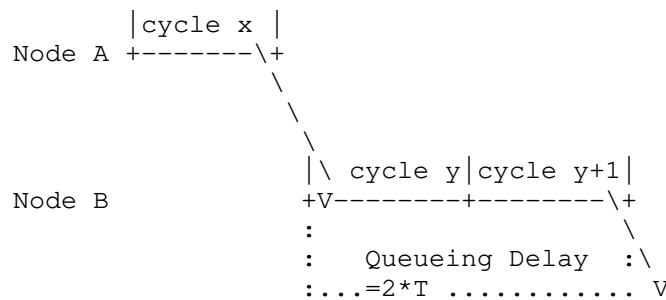


Figure 8: Single-Hop Queueing Delay

As Figure 8 shows, cycle x of Node A will be mapped into cycle y+1 of Node B as long as the last packet sent from A->B is received within the cycle y. If the last packet is re-sent out by B at the end of cycle y+1, then the largest single-hop queueing delay is  $2 \cdot T$ . Therefore the end-to-end queueing delay's upper bound is  $2 \cdot T \cdot H$ , where H is the number of hops.

If A did not forward the LDN packet from a prior LDN forwarder but is the actual traffic source, then the packet may have been delayed by a gate function before it was sent to B. The delay of this function is outside of scope for the LDN delay considerations. If B is not forwarding the LDN packet but the final receiver, then the packet may not need to be queued and released in the same fashion to the receiver as it would be queued/released to a downstream LDN node, so if a path has one source followed by N LDN forwarders followed by one receivers, this should be considered to be a path with N-1 LDN hops for the purpose of latency and jitter calculations.

##### 4.2. Jitter

Considering the simplest scenario one hop forwarding at first, suppose Node A is the upstream node of Node B, the packet sent from

Node A at cycle  $x$  will be received by Node B at cycle  $y$  as Figure 9 shows.

- The best situation is Node A sends packet at the end of cycle  $x$ , and Node B receives packet at the beginning of cycle  $y$ , then the delay is denoted by  $w$ ;
- The worst situation is Node A sends packet at the beginning of cycle  $x$ , and Node B receives packet at the end of cycle  $y$ , then the delay =  $w + \text{length of cycle } x + \text{length of cycle } y = w + 2 * T$ ;
- Hence the jitter's upper bound of this simplest scenario = worst case - best case =  $2 * T$ .

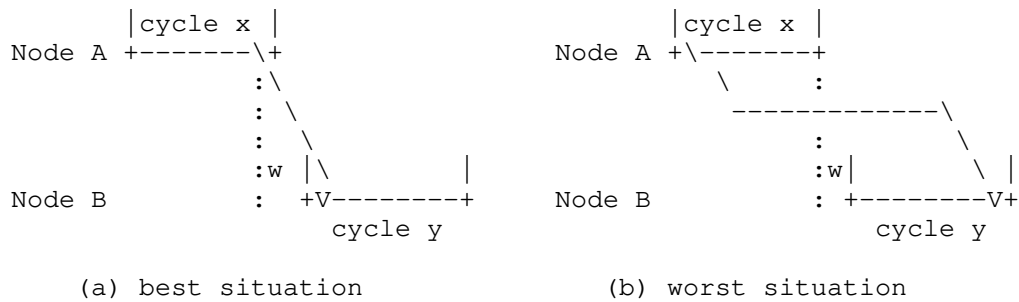
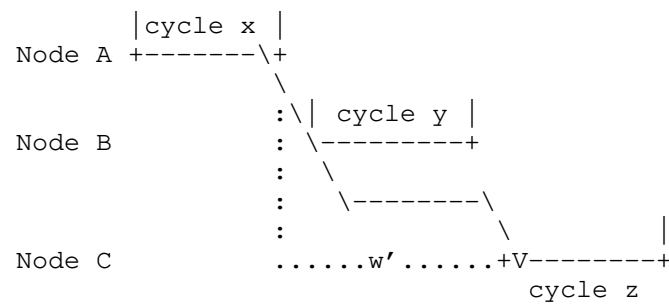


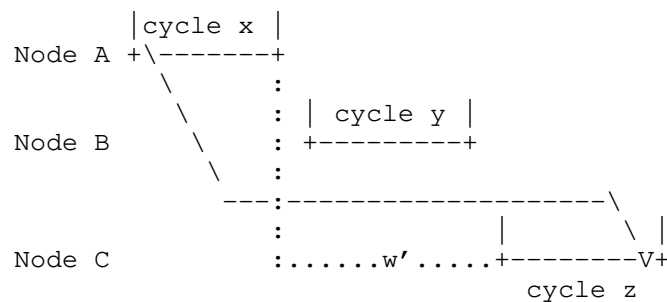
Figure 9: Jitter Analysis for One Hop Forwarding

Next considering two hops forwarding as Figure 10 shows.

- The best situation is Node A sends packet at the end of cycle  $x$ , and Node C receives packet at the beginning of cycle  $z$ , then the delay is denoted by  $w'$ ;
- The worst situation is Node A sends packet at the beginning of cycle  $x$ , and Node C receives packet at the end of cycle  $z$ , then the delay =  $w' + \text{length of cycle } x + \text{length of cycle } z = w' + 2 * T$ ;
- Hence the jitter's upper bound = worst case - best case =  $2 * T$ .



(a) best situation



(b) worst situation

Figure 10: Jitter Analysis for Two Hops Forwarding

And so on. For multi-hop forwarding, the end-to-end delay will increase as the number of hops increases, while the delay variation (jitter) still does not exceed  $2 \cdot T$ .

## 5. IANA Considerations

This document makes no request of IANA.

## 6. Security Considerations

Security issues have been carefully considered in [draft-ietf-detnet-security]. More discussion is TBD.

## 7. Acknowledgements

TBD.



## 8. Normative References

- [draft-ietf-detnet-architecture]  
"DetNet Architecture", <<https://datatracker.ietf.org/doc/draft-ietf-detnet-architecture/>>.
- [draft-ietf-detnet-dp-sol]  
"DetNet Data Plane Encapsulation",  
<<https://datatracker.ietf.org/doc/draft-ietf-detnet-dp-sol/>>.
- [draft-ietf-detnet-problem-statement]  
"DetNet Problem Statement",  
<<https://datatracker.ietf.org/doc/draft-ietf-detnet-problem-statement/>>.
- [draft-ietf-detnet-security]  
"DetNet Security Considerations",  
<<https://datatracker.ietf.org/doc/draft-ietf-detnet-security/>>.
- [draft-ietf-detnet-use-cases]  
"DetNet Use Cases", <<https://datatracker.ietf.org/doc/draft-ietf-detnet-use-cases/>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [scheduled-queues]  
"Scheduled queues, UBS, CQF, and Input Gates",  
<<http://www.ieee802.org/1/files/public/docs2015/new-nfinn-input-gates-0115-v04.pdf>>.

## Authors' Addresses

Li Qiang (editor)  
Huawei  
Beijing  
China

Email: [qiangli3@huawei.com](mailto:qiangli3@huawei.com)

Xuesong Geng  
Huawei  
Beijing  
China

Email: gengxuesong@huawei.com

Bingyang Liu  
Huawei  
Beijing  
China

Email: liubingyang@huawei.com

Toerless Eckert (editor)  
Huawei USA - Futurewei Technologies Inc.  
2330 Central Expy  
Santa Clara 95050  
USA

Email: tte+ietf@cs.fau.de

Liang Geng  
China Mobile  
Beijing  
China

Email: gengliang@chinamobile.com