Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 22, 2019

M. Chen
X. Geng
Huawei
Z. Li
China Mobile

October 19, 2018

Segment Routing (SR) Based Bounded Latency
draft-chen-detnet-sr-based-bounded-latency-00

Abstract

One of the goals of DetNet is to provide bounded end-to-end latency for critical flows. This document defines how to leverage Segment Routing (SR) to implement bounded latency. Specifically, the SR Identifier (SID) is used to specify transmission time (cycles) of a packet. When forwarding devices along the path follow the instructions carried in the packet, the bounded latency is achieved. This is called Cycle Specified Queuing and Forwarding (CSQF) in this document.

Since SR is a source routing technology, no per-flow state is maintained at intermediate and egress nodes, SR-based CSQF naturally supports flow aggregation that is deemed to be a key capability to allow DetNet to scale to large networks.

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 April 22, 2019.

Table of Contents

1. Introduction

Deterministic Networking (DetNet) [I-D.ietf-detnet-architecture] is defined to provide end-to-end bounded latency and extremely low packet loss rates for critical flows. For a specific path, the end-to-end latency consists of two parts: 1) the accumulated latency on the wire, 2) the accumulated latency of nodes along the path. The former can be considered as constant once the path has been determined. The latter is contributed by the latency within each node along the path. So, to guarantee the end-to-end bounded latency, control the bounded latency within a node is the key. If every node along the path can guarantee bounded latency, then end-to-end bounded latency can be achieved.

[I-D.finn-detnet-bounded-latency] gives a framework that describes how bounded latency and zero congestion loss are achieved. It introduces a parameterized timing model that can be used by DetNet solutions by selecting a corresponding Quality of Service (QoS) algorithm and resource reservation algorithm to achieve the bounded latency and zero congestion loss goal.

This document defines how to leverage Segment Routing (SR) [RFC8402] to implement bounded latency. Specifically, the SR Identifier (SID) is used to carry and specify the "sending time" (cycle) of a packet, and ensure that the packet will be transmitted in that specified sending cycle in order to achieve the bounded latency. This is called Cycle Specified Queuing and Forwarding (CSQF) in this document.

2. Cycle Specified Queuing and Forwarding

2.1. CSQF Basic Concepts

By specifying the sending cycle of a packet at a node and making sure that the packet will be transmitted in that cycle, CSQF can achieve bounded latency within the node. By specifying the sending cycle at every node along a path, the end-to-end bounded latency can be achieved.

To support CSQF, similar to Cyclic Queuing and Forwarding (CQF) [IEEE802.1Qch], the sending time of an output interface of a node is divided into a series of equal time intervals with the duration of T. Each time interval is called a "cycle", and each cycle corresponds to a queue. During a cycle, only the corresponding queue is open and all the packets in that queue will be transmitted. CSQF can not only control the bounded latency at every node along a path, but regulate the traffic at each node as planned. Therefore, no congestion will occur.

Figure 1 provides an overview of CSQF.

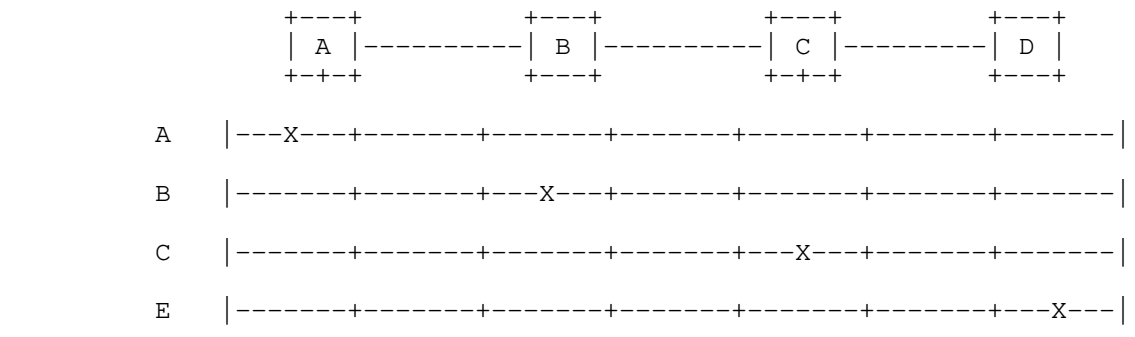


Figure 1: CSQF Overview

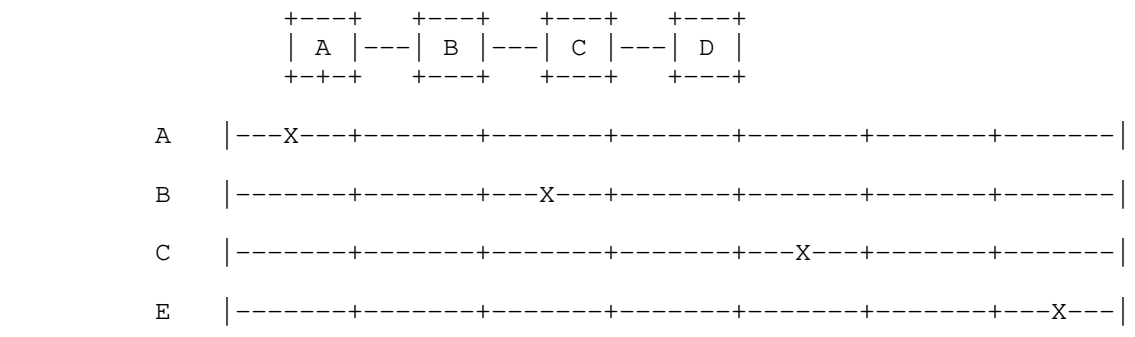


Figure 1: CSQF Overview

CSQF has the following characteristics:

- The sending time (cycle) of a packet at each node along a path is specified so that the packet will be transmitted in the specified cycles, hence to guarantee the end-to-end bounded latency.

- The specified cycles are calculated by fully considering the link delay, processing delay and the available cycle resources,

resulting in no bandwidth waste and no congestion (cycle-based traffic regulation).

- Segment routing (SR) is used. Specifically, a SID is used to indicate in which cycle and to which output interface that a packet is specified to transmit, and an SR SID list is used to carry the specified cycles along a path. With SR, there is no per-flow states maintained at the intermediate and egress node. As a result, scalability is greatly improved compared to a solution that maintains flow state at each hop.

- Flow aggregation is naturally supported by introducing SR and cycle-based scheduling.

2.2. CSQF Queuing Model

In Cyclic Queuing and Forwarding (CQF) [IEEE802.1Qch], time is divided into numbered time intervals, and each time interval is called a cycle; the critical traffic is then transmitted and queued for transmission along a path in a cyclic manner. With CQF, the delays experienced by a given packet are as follows:

- The maximum end-to-end delay = (N+1) * T;

- The minimum end-to-end delay = (N-1) * T;

- Where the N is the number of hops and T is the duration of the cycle.

CQF assumes that a packet is transmitted from an upstream node in a cycle and the packet must be received at the downstream node in the same cycle, and it must be transmitted in the next cycle to the nexthop node. This assumption leads to very low bandwidth utilization when the link delay, processing delay, etc., factors cannot be considered as trivial. To guarantee this assumption, more bandwidth has to be reserved as a guard band for each cycle, and the effective bandwidth for DetNet service will be greatly reduced.

CSQF improves on CQF by explicitly specifying the sending cycles at every node along the path. This relieves the limitation that the sending (at the upstream node) and receiving (at the downstream node) have to be in the same cycle. For CSQF, the cycle to use depends on traffic planning and path calculation. The path calculation will consider the available cycle resources, bandwidth, and delay constraints.

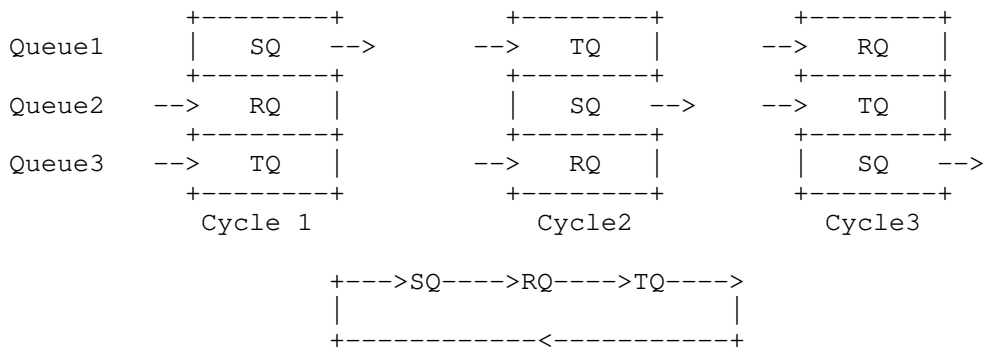


Figure 2: CSQF Queuing Model

For CSQF, three queues (in theory, two or more queues work as well) for each output interface are used. During a particular cycle, only one queue is open and the packets in that queue will be transmitted. This queue is called the sending queue (SQ). The other two queues are closed and can enqueue packets. One of them is called the receiving queue (RQ). The third queue is called the tolerating queue (TQ).

The RQ is used for receiving the packets that are expected to be transmitted in the next cycle. The TQ is used for tolerating the packets that come a bit early due to processing delay variation (processing jitter) or other reasons (e.g., packets are not transmitted as required by the traffic specification). Both RQ and TQ can have the capability to absorb a certain amount of processing jitter and traffic bursts. The upper bound of the absorbing capacity is 2T. In order to increase the jitter/burst absorbing capacity, a four or more-queue model can be used. If the processing delay and traffic bursts are small, two-queue model works as well.

The roles of the three queues are not fixed, and on the contrary, they rotate with each cycle change. As showed in Figure 2, during cycle 1, queue 1 is SQ, queue 2 is RG and queue 3 is TQ; during cycle 2, queue 1 is TQ, queue 2 is SQ and queue 3 is RQ, during cycle 3, queue 1 is RQ, queue 2 is TQ and queue 3 is SQ. That means, for a particular queue, its role will rotate as "...->SQ->RQ->TQ->SQ->...", the starting role of a queue can be any one of the three roles.

In CSQF, a cycle corresponds to a queue. There are several ways to do cycle to queue mapping. The simplest mapping between cycles and queues is 1:1 mapping. There could be N:1 mapping, but that requires more identifiers, which in the case of segment routing, would require

more SIDs. This document does not specify which mapping should be used. The mapping choice is left to the operator.

2.3. CSQF Timing Model

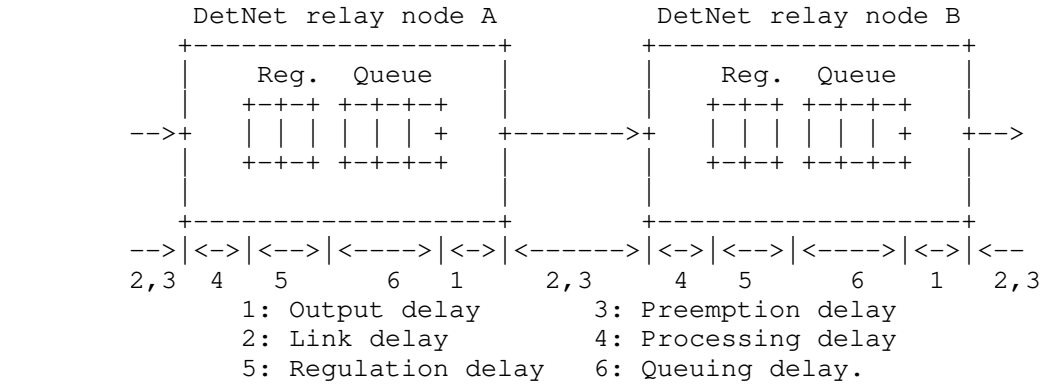


Figure 3: Timing model for DetNet

The DetNet timing model in Figure 3 is defined in [I-D.finn-detnet-bounded-latency]. It details the delays that a packet can experience from hop to hop. There are six delays, the detailed explanation of which can be found in [I-D.finn-detnet-bounded-latency]. This document simplifies the above model as follows:

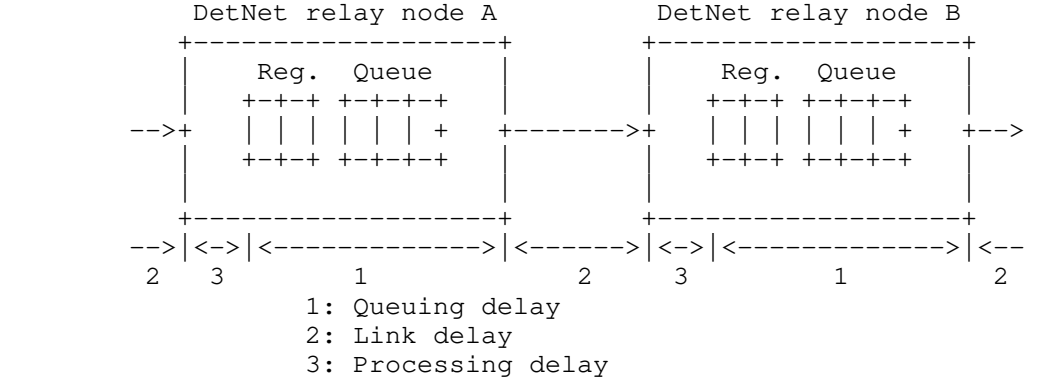


Figure 4: Simplified Timing model for DetNet

In this simplified timing model, only three delays are defined. The queuing delay in this new model includes the output delay, regulation delay, and queuing delay that are defined in the DetNet timing model (Figure 3). The link delay defined in this document includes the

link delay and the preemption delay defined in [I-D.finn-detnet-bounded-latency]. The processing delay is the same as defined in [I-D.finn-detnet-bounded-latency].

To further simplify the model, it assumes that the link delay only depends on the distance of the link. Once the DetNet path has been determined, the link delay can be considered as constant. The processing delay and queuing delay are variable but have their upper bounds.

For the processing delay, there are two bounds: minimum processing delay (Min-P-Delay) and maximum processing delay (Max-P-Delay).

- Thus, the maximum processing jitter (Max-P-Jitter) = Max-P-Delay - Min-P-Delay.

As described in Section 2.2, both the RQ and TQ can be used for absorbing processing jitter, and the upper bound of the absorbing capacity is 2T. So, if the processing jitter is less than 2T, the three-queue model can work. Otherwise, more buffer is needed to absorb the jitter, through increasing the duration of the cycle or by adding more queues. Increasing the duration of the cycles is equivalent to increasing the depth of the queues (adding more buffer for each queue).

With above, for CSQF, the delays experienced by a given packet are as follows:

- The maximum end-to-end delay = Link delay + N * (Max-P-Delay + 2T);

- The maximum end-to-end jitter = 2T;

- Where N is the number of hops and T is the duration of a cycle.

2.4. Congestion Protection and Resource Reservation

Congestion protection is the key for bounded latency and zero congestion loss. An essential component of DetNet is Traffic Engineering (TE), so that dedicated resources can be reserved for the exclusive use of DetNet flows. To avoid congestion, two or more flows must be prevented from contending for the same resource. For normal TE, the critical resource is bandwidth, but in the case of CSQF, the critical resource is interface occupation time. Bandwidth is an average value, which can generally guarantee the quality of service generally, but bursts and congestion may still occur. By comparison, the interface occupation time is an absolute value, which can avoid packet packets conflicting for the same resource by

controller computation and time allocation for different flows. The unit of time allocation is the cycle, and a Traffic Specification, the flow transmission description, is necessary for the computation.

CSQF uses segment routing SIDs to carry the time allocation information (the cycle), and it ensures that a node can schedule different packets without conflict and forward the packets at the proper time. The resource reservation is not explicitly implemented by a control plane protocol, such as Resource Reservation Protocol - Traffic Engineering (RSVP-TE) or Stream Reservation Protocol (SRP). Rather, it is guaranteed by the SR controller, which maintains the status of different flows and time occupation of all the network devices in the domain. This is called the Virtual Resource Reservation (VRR) in this document.

2.5. An Example of CSQF

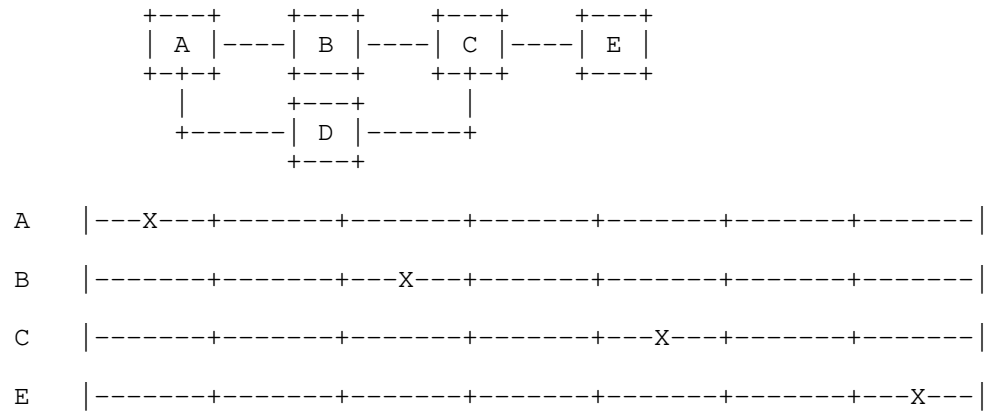


Figure 5: CSQF Example

As showed in Figure 5, there is a DetNet path (A->B->C->E), and a packet (X) is expected to be transmitted in cycle 1 at node A, in cycle 3 at node B, in cycle 5 at node C and in cycle 7 at node E. A cycle list <1, 3, 5, 7> is attached to the packet, and the packet will be transmitted along the path as the specified cycles.

Given the topology as above, assume the duration of a cycle is 10us; the link delays between nodes are the same (e.g., 100us); the minimum

processing delay at each node = 10us, the maximum processing delay at each node is 20us, so the maximum processing jitter is 10us.

For a given packet that is transmitted along the path(A->B->C->D->E), the experienced maximum end-to-end delay is:

(N-1) * link delay + N * (maximum processing delay + 2T)

= 3*100 + 4* 40

= 460 (us)

The maximum end-to-end jitter is always 2T (20us).

3. Segment Routing Extensions for CSQF

This document defines a new segment that is called a Cycle Segment, which is used to identify a cycle. A Cycle Segment is a local segment and is allocated from the Segment Routing Local Block (SRLB)[RFC8402].

A Cycle Segment has two meanings: 1) identify an interface/link, just like the adjacency segment does; 2) identify a cycle of the interface/link. To specify to which interface and in which cycle a packet should be transmitted, it just needs to attach a Cycle Segment to the packet. By attaching a list of Cycle Segments to a packet, it can not only implement the explicit route of the packet that is required by DetNet [I-D.ietf-detnet-architecture], but also specify the sending cycle at each node along the path without maintaining per-flow states at the intermediate and egress nodes. Hence, it naturally supports flow aggregation, and that allows DetNet to support large number of DetNet flows and scale to large networks.

Normally, several SR SIDs are required to be allocated for each CSQF capable interface. How many SIDs are allocated depends on how many cycles are used. Given a three-queue model and a 1:1 cycle to queue mapping is used, three SIDs will be allocated for each CSQF capable interface. For example, given node A, SR-MPLS SIDs 1001, 1002, and 1003 are allocated to one of its interfaces. SID 1001 identifies cycle 1, SID 1002 identifies cycle 2, SID 1003 identifies cycle 3.

The SR [RFC8402] can be instantiated on various data planes. There are two data-plane instantiations of SR: SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6). Both SR-MPLS and SRv6 SIDs can be used for CSQF cycle identification. The mapping (IGP extensions) between a cycle and a SID will be defined in a separate document.

4. IANA Considerations

This document makes no request of IANA.

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

5. Security Considerations

6. Acknowledgements

The authors would like to thank Andrew G. Malis, Norman Finn for his review, suggestion and comments to this document.

7. References

7.1. Normative References

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

7.2. Informative References

[I-D.finn-detnet-bounded-latency]
Finn, N., Boudec, J., Mohammadpour, E., Varga, B., and J. Farkas, "DetNet Bounded Latency", draft-finn-detnet-bounded-latency-01 (work in progress), July 2018.

[I-D.geng-detnet-conf-yang]
Geng, X., Chen, M., Li, Z., and R. Rahman, "DetNet Configuration YANG Model", draft-geng-detnet-conf-yang-05 (work in progress), October 2018.

[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-02 (work in progress), March 2018.

[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.

[IEEE802.1Qch]
"IEEE, "Cyclic Queuing and Forwarding (IEEE Draft P802.1Qch)", 2017, <http://www.ieee802.org/1/files/private/ch-drafts/>.", 2016.

[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

Mach(Guoyi) Chen
Huawei

Email: mach.chen@huawei.com

Xuesong Geng
Huawei

Email: gengxuesong@huawei.com

Zhenqiang Li
China Mobile

Email: lizhenqiang@chinamobile.com

DetNet
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2019

N. Finn
Huawei Technologies Co. Ltd
J-Y. Le Boudec
E. Mohammadpour
EPFL
B. Varga
J. Farkas
Ericsson

July 2, 2018

DetNet Bounded Latency
draft-finn-detnet-bounded-latency-01

Abstract

This document presents a parameterized timing model for Deterministic Networking so that existing and future standards can achieve bounded latency and zero congestion loss.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 3, 2019.

Copyright Notice

Table of Contents

1. Introduction

The ability for IETF Deterministic Networking (DetNet) or IEEE 802.1 Time-Sensitive Networking (TSN) 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.

Rather than defining, in great detail, specific mechanisms to be used to control packet transmission at each output port, this document presents a timing model for sources, destinations, and the network nodes that relay packets. The parameters specified in this model:

- Characterize a DetNet flow in a way that provides externally measureable 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.

- Enable resonably 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 devince, 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 QoS algorithms 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 a set of requirements for resource reservation algorithms and for QoS algorithms that, if met, will enable them to work together.

2. 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.

3. Terminology and Definitions

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

4. DetNet bounded latency model

4.1. Flow creation

The bounded latency model assusmes the use of the following paradigm for provisioning a particular DetNet flow:

1. Perform any onfiguration required by the relay systems in the network for the classes of service to be offered, including one or more classes of DetNet service. This configuration is general; it is not tied to any particular flow.

2. Characterize the DetNet flow in terms of limitations on the sender Section 8.1 and flow requirements Section 8.2.

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 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 QoS algorithms 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 with 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. How the path to be taken by a DetNet flow is chosen is not considered in this document.

4.2. End-to-end model

[Suggestion: This is the introduction to network calculus. The starting point is a model in which a relay system is a black box.]

4.3. Relay system model

[NWF I think that at least some of this will be useful. We won't know until we see what J-Y has to say in Section 4.2. I'm especially interested in whether J-Y thinks that the "output delay" in Figure 1 is useful in determining the number of buffers needed in the next hop. It is possible that we can define the parameters we need without this section.]

In Figure 1 we see a breakdown of the per-hop latency experienced by a packet passing through a relay system, in terms that are suitable for computing both hop-by-hop latency and per-hop buffer requirements.

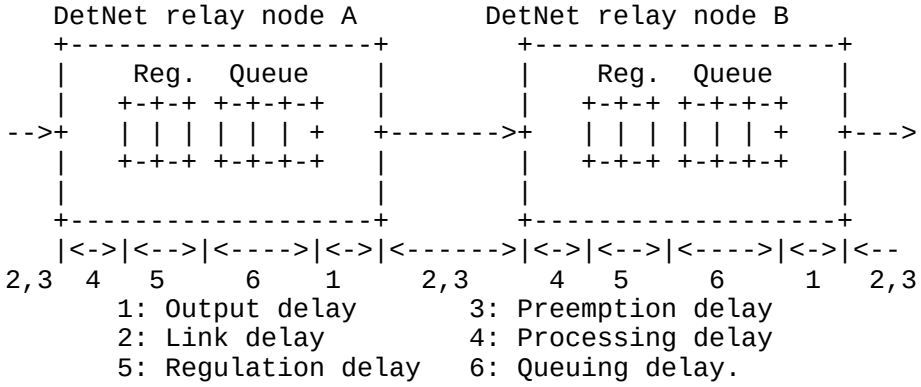


Figure 1: Timing model for DetNet or TSN

In Figure 1, we see two DetNet relay 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 five 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 (e.g. [IEEE8023br] preemption) in order to transmit another packet or packets, an arbitrary delay can result.

4. Processing delay
This delay covers the time from the reception of the last bit of the packet to that packet being eligible, if there were no other packets in the queue, for selection for output. This delay can be variable, and depends on the details of the operation of the forwarding node.

5. Regulation delay
This is the time spent from the insertion of the 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 delay
This is the time spent for a packet from being declared eligibile 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:

- 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.

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

5. Computing End-to-end Latency Bounds

End-to-end latency bounds can be computed using the delay model in Section 4.3. Here it is important to be aware that for several queuing 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

```
end_to_end_latency_bound = non_queuing_latency + queuing_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_queuing_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 node at the h-th hop but not on the T-SPEC of this detnet flow. Then set non_queuing_latency = the sum of per-hop_non_queuing_latency[h] over all hops h.

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

For several queuing mechanisms, queuing_latency is less than the sum of upper bounds on the queuing delays (5,6) at every hop. Section 5.1 gives such practical computation examples.

For other queuing mechanisms the only available value of queuing_latency is the sum of the per-hop queuing delay bounds. In such cases, the computation of per-hop queuing 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 node.

5.1. Examples of Computations

5.1.1. Per-flow queuing

[[JYLB: THIS IS WHERE DETAILS OF END-TO-END LATENCY COMPUTATION ARE GIVEN FOR PER-FLOW QUEUING]]

5.1.2. Time-Sensitive Networking with Asynchronous Traffic Shaping

Figure 2 shows an example of a network with 5 nodes, which have the queuing model as Section 7.3. An end-to-end delay bound for flow f of a given AVB class (A or B), traversing from node 1 to 5, is calculated as following:

```
end_to_end_latency_bound_of_flow_f = C12 + C23 + C34 + S4
```

In the above formula, Cij is a bound on the aggregate response time of the AVB FIFO queue with CBS (Credit Based Shaper) in node i and interleaved regulator of node j, and S4 is a bound on the response time of the AVB FIFO queue with CBS in node 4 for flow f. In fact, using the delay definitions in Section 4.3, Cij is a bound on sum of the delays 1,2,3,6 of node i and 4,5 of node j. Similarly, S4 is a bound on sum of the delays 1,2,3,6 of node 4. The detail of calculation for the these response time bounds can be found in [TSNwithATS].

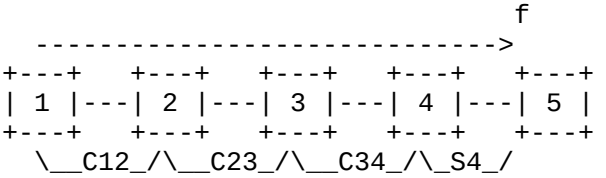


Figure 2: End-to-end latency computation example

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].

6. 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.

6.1. A General Formula

To avoid congestion losses, an upper bound on the backlog present in the queue of Figure 1 must be computed during path computation. This bound depends on the set of flows that use this queue, 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

- nb_classes be the number of classes of traffic that may use this output port
- 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.
- nb_input_ports be the number input ports that send traffic of any class to this output port
- 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 ouput port.

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

```
backlog_bound = ( nb_classes + nb_input_ports ) *
max_packet_length + total_in_rate* max_delay45
```

7. Queuing model

[[JYLB: THIS IS WHERE DETAILS OF END-TO-END LATENCY COMPUTATION ARE GIVEN FOR PER-FLOW QUEUING AND FOR TSN WITH ATS]]

7.1. Queuing data model

Sophisticated QoS 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 instantiated hierarchically above the "Layer 2" queuing mechanisms, among which packets compete for opportunities to be transmitted on a physical (or sometimes, logical) medium. These "Layer 2 queuing mechanisms" are not the province solely of bridges; they are an essential part of any DetNet relay node. As illustrated by numerous implementation examples, the "Layer 3" some of mechanisms described in documents such as [RFC7806] are often integrated, in an implementation, with the "Layer 2" mechanisms also implemented in the same system. 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 (very simple) model for the flow of packets through the queues of an IEEE 802.1Q bridge. Packets are assigned to a class of service. The classes of service are mapped to some number of physical FIFO queues. IEEE 802.1Q allows a maximum of 8 classes of service, but it is more common to implement 2 or 4 queues on most ports.

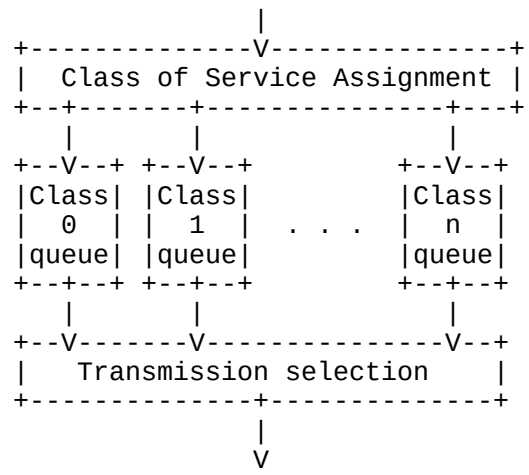


Figure 3: IEEE 802.1Q Queuing Model: Data flow

Some relevant mechanisms are hidden in this figure, and are performed in the "Class n queue" box:

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

The Class of Service Assignment function can be quite complex, since the introduction of [IEEE802.1Qci]. In addition to the Layer 2 priority expressed in the 802.1Q VLAN tag, a bridge can utilize any of the following information to assign a packet to a particular class of service (queue):

- Input port.
- Selector based on a rotating schedule that starts at regular, time-synchronized intervals and has nanosecond precision.
- MAC addresses, VLAN ID, IP addresses, Layer 4 port numbers, DSCP. (Work items expected to add MPC and other indicators.)
- The Class of Service Assignment function can contain metering and policing functions.

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

7.2. IEEE 802.1 Queuing Model

7.2.1. Queuing Data Model with Preemption

Figure 3 must be modified if the output port supports preemption ([IEEE8021Qbu] and [IEEE8023br]). This modification is shown in Figure 4.

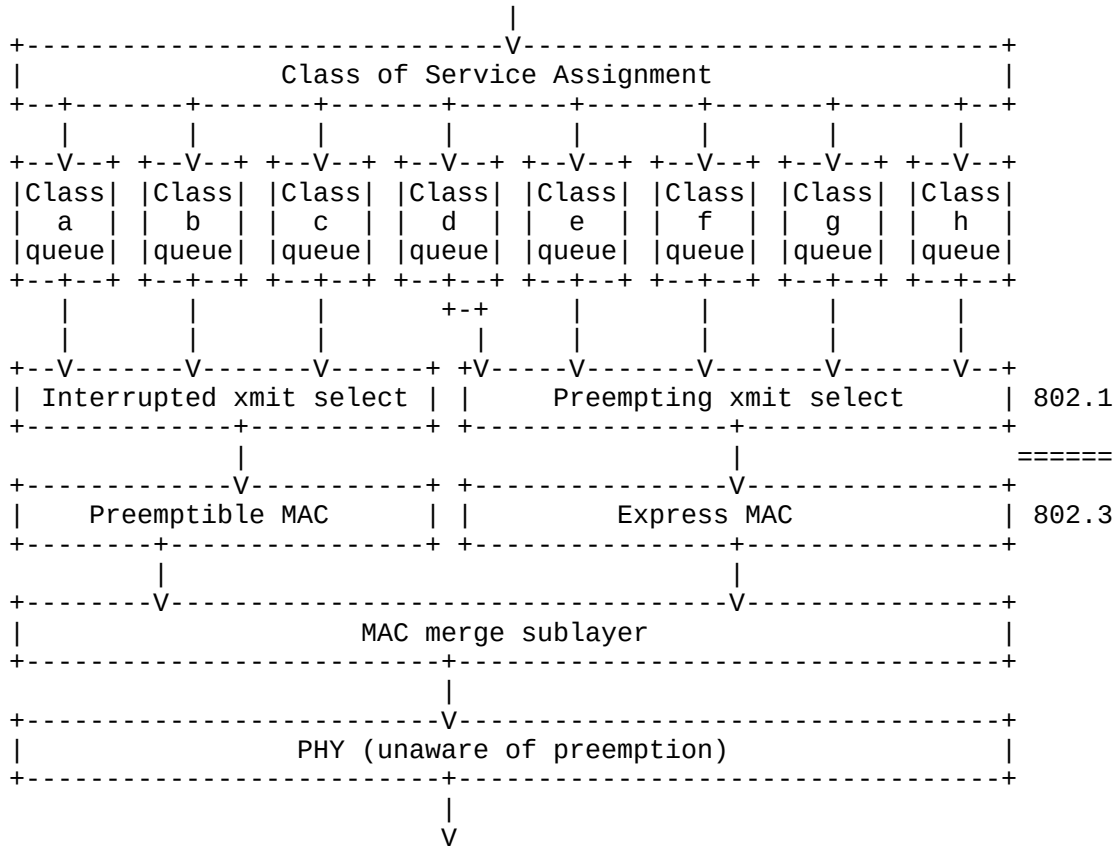


Figure 4: IEEE 802.1Q Queuing Model: Data flow with preemption

From Figure 4, we can see that, in the IEEE 802 model, the preemption feature 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. In Figure 4, the classes of service are marked "a, b, ..." instead of with numbers, in order to avoid any implication about which numeric Layer 2 priority values correspond to preemptible or preempting queues. Although it shows

three queues going to the preemptible MAC/PHY, any assignment is possible.

7.2.2. Transmission Selection Model

In Figure 5, we expand the "Transmission selection" function of Figure 4.

Figure 5 does NOT show the data path. It shows an example of a configuration of the IEEE 802.1Q transmission selection box shown in Figure 3 and Figure 4. Each queue m presents a "Class m Ready" signal. These signals go through various logic, filters, and state machines, until a single queue’s "not empty" signal is chosen for presentation to the underlying MAC/PHY. When the MAC/PHY is ready to take another output packet, then a packet is selected from the one queue (if any) whose signal manages to pass all the way through the transmission selection function.

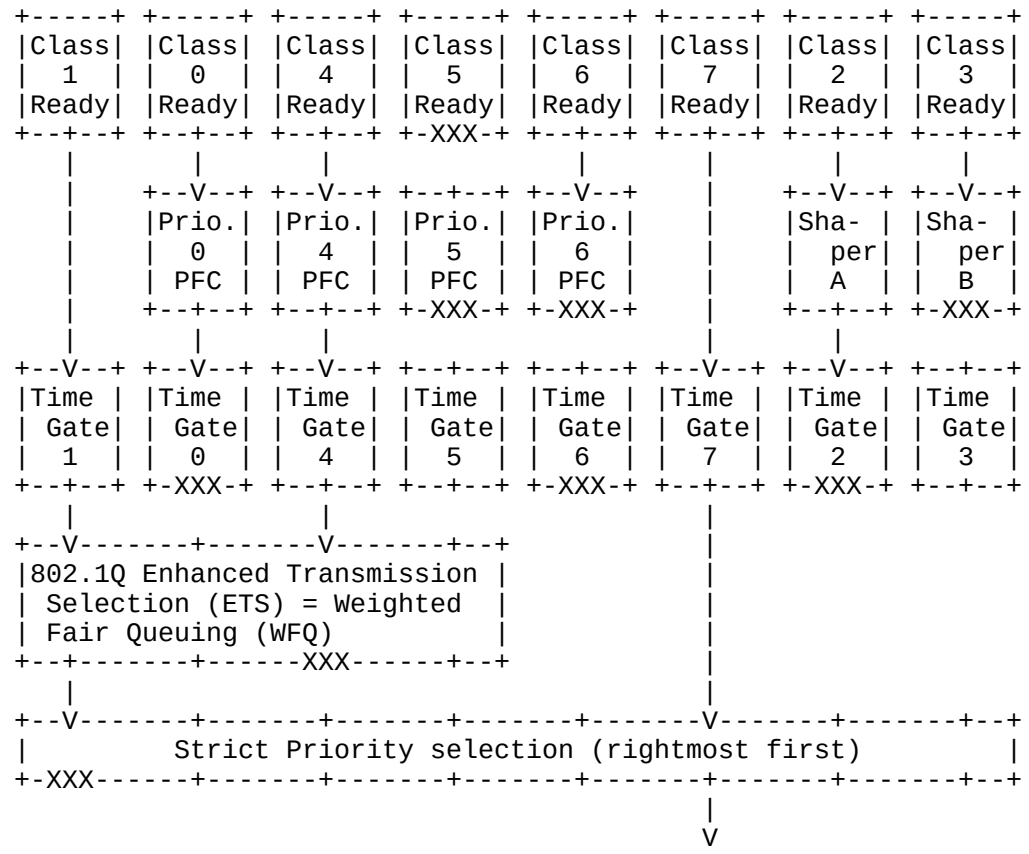


Figure 5: 802.1Q Transmission Selection

The following explanatory notes apply to Figure 5

- The numbers in the "Class n Ready" boxes are the values of the Layer 2 priority that are assigned to that Class of Service in this example. The rightmost CoS is the most important, the leftmost the least. Classes 2 and 3 are made the most important, because they carry DetNet flows. It is all right to make them more important than the priority 7 queue, which typically carries critical network control protocols such as spanning tree or IS-IS, because the shaper ensures that the highest priority best-effort queue (7) will get reasonable access to the MAC/PHY. Note that Class 5 has no Ready signal, indicating that that queue is empty.

- Below the Class Ready signals are shown the Priority Flow Control gates (IEEE Std 802.1Qbb-2011 Priority-based Flow Control, now [IEEE8021Q] clause 36) on Classes of Service 1, 0, 4, and 5, and

two 802.1Q shapers, A and B. Perhaps shaper A conforms to the IEEE Std 802.1Qav-2009 (now [IEEE8021Q] clause 34) credit-based shaper, and shaper B conforms to [IEEE8021Qcr] Asynchronous Traffic Shaper. Any given Class of Service can have either a PFC function or a shaper, but not both.

- Next are the IEEE Std 802.1Qbv time gates ([IEEE8021Qbv]). Each one of the 8 Classes of Service has a time gate. The gates are controlled by a repeating schedule that restarts periodically, and can be programmed to turn any combination of gates on or off with nanosecond precision. (Although the implementation is not necessarily that accurate.)

- Following the time gates, any number of Classes of Service can be linked to one ore more instances of the Enhanced Transmission Selection function. This does weighted fair queuing among the members of its group.

- A final selection of the one queue to be selected for output is made by strict priority. Note that the priority is determined not by the Layer 2 priority, but by the Class of Service.

- An "XXX" in the lower margin of a box (e.g. "Prio. 5 PFC" indicates that the box has blocked the "Class n Ready" signal.

- IEEE 802.1Qch Cyclic Queuing and Forwarding [IEEE802.1Qch] is accomplished using two or three queues (e.g. 2 and 3 in the figure), using sophisticated time-based schedules in the Class of Service Assignment function, and using the IEEE 802.1Qbv time gates [IEEE8021Qbv] to swap between the output buffers.

7.3. Time-Sensitive Networking with Asynchronous Traffic Shaping

Consider a network with a set of nodes (switches 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 as 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 6. 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].

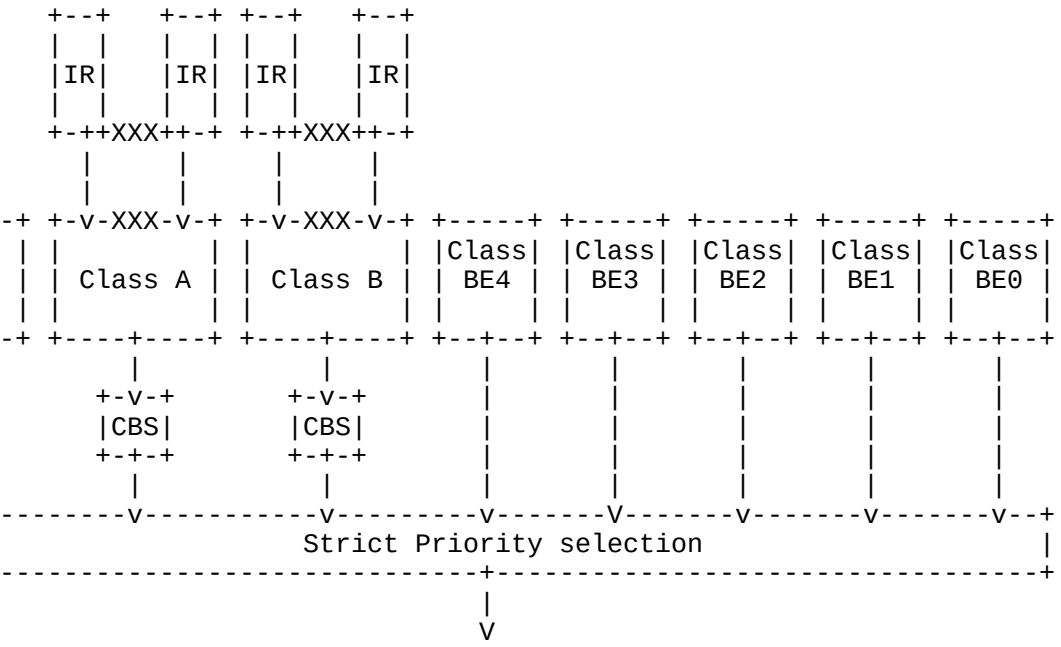


Figure 6: Architecture of one TSN node output port with interleaved regulators (IRs)

The CBFS includes two 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 an affine arrival curve r t + b in each node, i.e. the amount of bits entering a node within a time interval t is bounded by r t + b.

[[EM: THE FOLLOWING PARAGRAPH SHOULD BE ALIGNED WITH Section 8.2.]]

Additionally, it is assumed that flows are regulated at their source, according to either leaky bucket (LB) or length rate quotient (LRQ). The LB-type regulation forces flow f to conform to an arrival curve r_f t+b_f . The LRQ-type regulation with rate r_f ensures that the time separation between two consecutive packets of sizes l_n and l_n+1 is at least l_n/r_f. Note that if flow f is LRQ-regulated, it satisfies an arrival curve constraint r_f t + L_f where L_f is its maximum packet size (but the converse may not hold). For an LRQ regulated flow, b_f = L_f. At the source hosts, the traffic satisfies its regulation constraint, i.e. the delay due to interleaved regulator at hosts is ignored.

At each switch 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 regulation constraints; it is released at the earliest time at which this is possible without violating the constraint. The regulation type and parameters for a flow are the same at its source and at all switches along its path.

7.4. Other queuing models, e.g. IntServ

[[NWF More sections that discuss specific models]]

8. Parameters for the bounded latency model

8.1. Sender parameters

8.2. Relay system parameters

[[NWF This section talks about the paramters that must be passed hop-by-hop (T-SPEC? F-SPEC?) by a resoure reservation protocol.]]

9. References

9.1. Normative References

[I-D.ietf-detnet-architecture]
Finn, N. and P. Thubert, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-00 (work in progress), September 2016.

[I-D.ietf-detnet-dp-alt]
Korhonen, J., Farkas, J., Mirsky, G., Thubert, P., Zhuangyan, Z., and L. Berger, "DetNet Data Plane Protocol and Solution Alternatives", draft-ietf-detnet-dp-alt-00 (work in progress), October 2016.

[I-D.ietf-detnet-use-cases]
Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-16 (work in progress), May 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>.

[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>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <https://www.rfc-editor.org/info/rfc4364>.

[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

[IEEE802.1Qch]
IEEE, "IEEE Std 802.1Qch-2017 IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks Amendment 29: Cyclic Queuing and Forwarding (amendment to 802.1Q-2014)", 2017, <http://www.ieee802.org/1/files/private/ch-drafts/>.

[IEEE802.1Qci]
IEEE, "IEEE Std 802.1Qci-2017 IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks - Amendment 30: Per-Stream Filtering and Policing", 2017, <http://www.ieee802.org/1/files/private/ci-drafts/>.

[IEEE8021Q]
IEEE 802.1, "IEEE Std 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>.

[IEEE8021Qbu]
IEEE, "IEEE Std 802.1Qbu-2016 IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks - Amendment 26: Frame Preemption", 2016, <http://standards.ieee.org/getieee802/download/802.1Qbu-2016.zip>.

[IEEE8021Qbv]
IEEE 802.1, "IEEE Std 802.1Qbv-2015: IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic", 2015, <http://standards.ieee.org/getieee802/download/802.1Qbv-2015.zip>.

[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 802.3-2015: IEEE Standard for Local and metropolitan area networks - Ethernet", 2015, <http://standards.ieee.org/getieee802/download/802.3-2015.zip>.

[IEEE8023br]
IEEE 802.3, "IEEE Std 802.3br-2016: IEEE Standard for Local and metropolitan area networks - Ethernet - Amendment 5: Specification and Management Parameters for Interspersing Express Traffic", 2016, <http://standards.ieee.org/getieee802/download/802.3br-2016.pdf>.

[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

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

Email: jean-yves.leboudec@epfl.ch

Ehsan Mohammadpour
EPFL
IC Station 14
Lausanne EPFL 1015
Switzerland

Email: ehsan.mohammadpour@epfl.ch

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

Email: balazs.a.varga@ericsson.com

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

Email: janos.farkas@ericsson.com

DetNet
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2019

J. Korhonen, Ed.

B. Varga, Ed.
Ericsson

October 21, 2018

DetNet IP Data Plane Encapsulation
draft-ietf-detnet-dp-sol-ip-01

Abstract

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

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 April 24, 2019.

Table of Contents

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 layers: a service layer and a transport layer. The service layer is used to provide DetNet service protection and reordering. The transport layer is used to provides congestion protection (low loss, assured latency, and limited reordering). As no DetNet specific headers are added to support IP DetNet flows, only the transport layer functions are supported using the IP DetNet 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 7 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] the reader is assumed to be familiar with that document.

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.

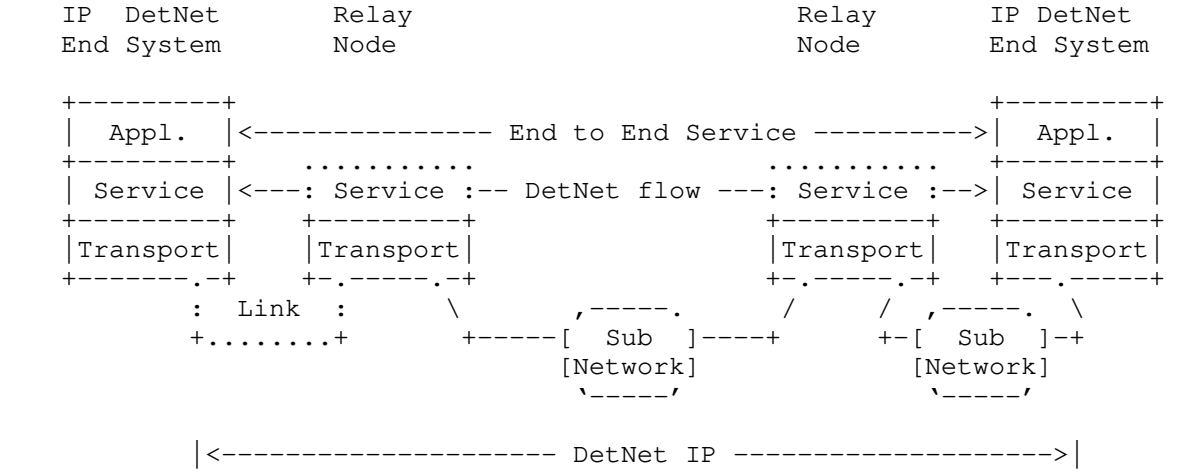


Figure 1: A Simple DetNet (DN) Enabled IP Network

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 transport 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 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 IP DetNet flows to TSN streams and TSN protection mechanisms is covered in Section 7.

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

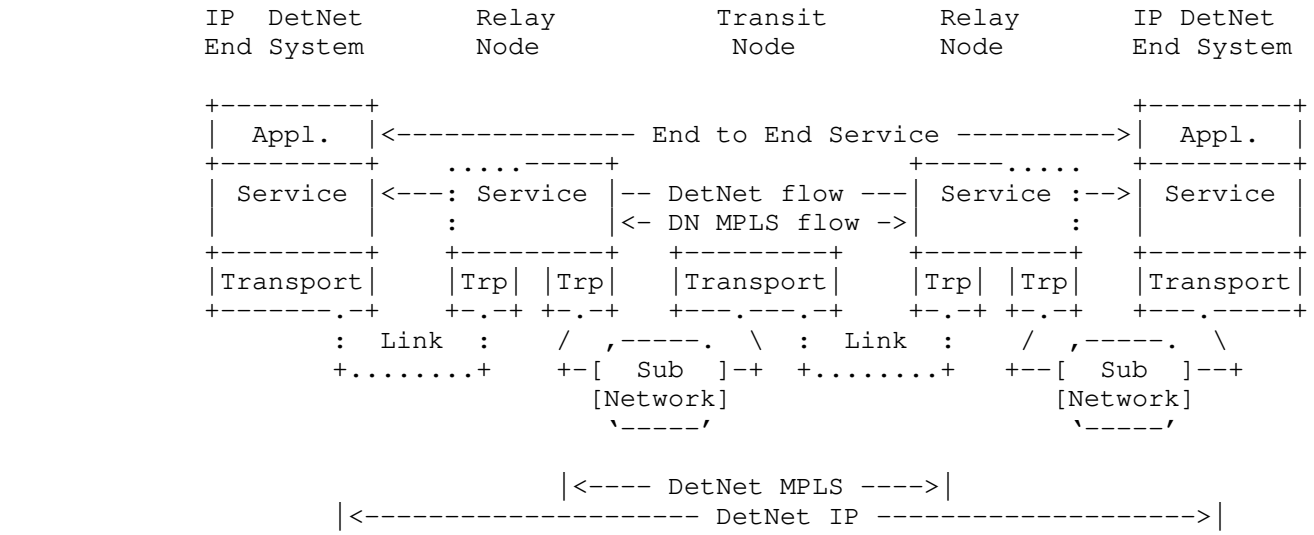


Figure 2: DetNet (DN) IP Over MPLS Network

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 [I-D.ietf-detnet-dp-sol-mpls] to map each DetNet flow to MPLS LSPs. While not shown, relay nodes can provide service layer functions such as PREOF over the MPLS transport layer, 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 [I-D.ietf-detnet-dp-sol-mpls] for details on the mapping of IP flows to MPLS as well as general support for DetNet services using MPLS.

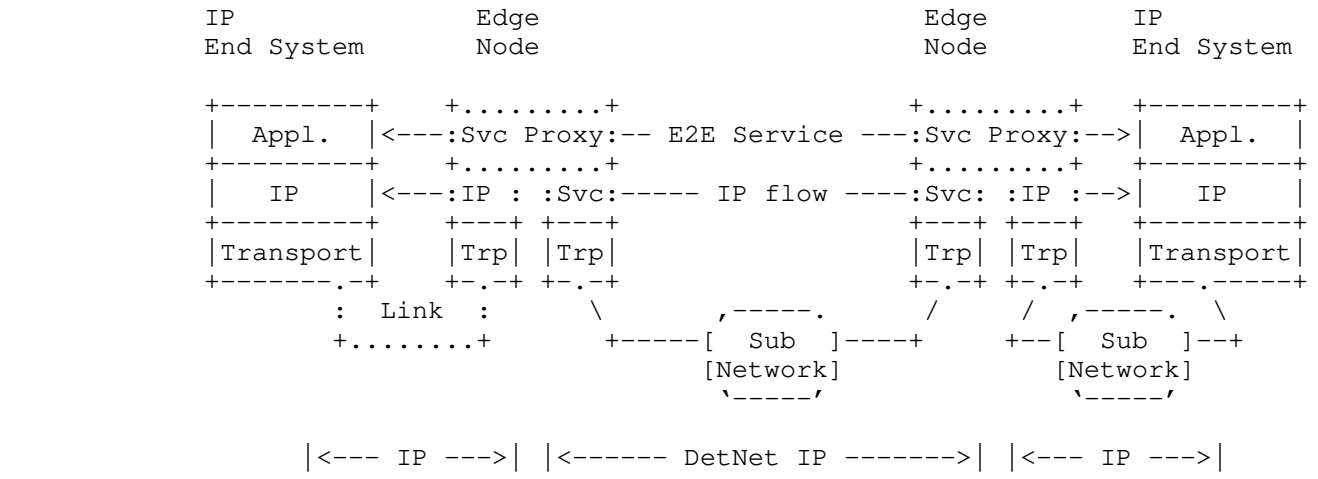


Figure 3: Non-DetNet aware IP end systems with IP DetNet Domain

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 miss-ordering 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 transporting 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 MUST allow 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.

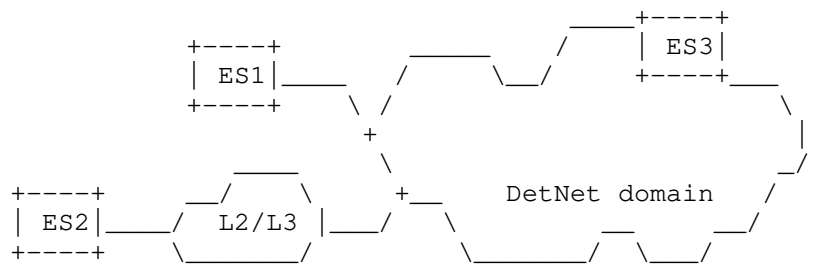


Figure 4: Connection types of L3 end systems

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.

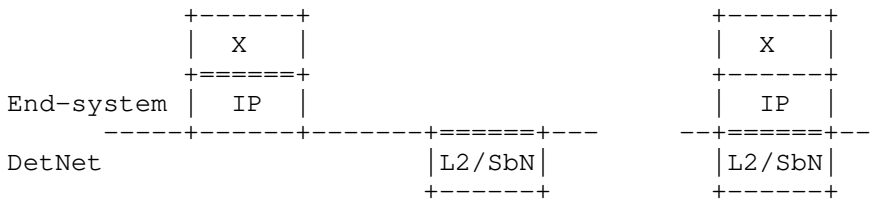


Figure 5: Encapsulation of DetNet Routing in simplified IP service L3 end-systems

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

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.

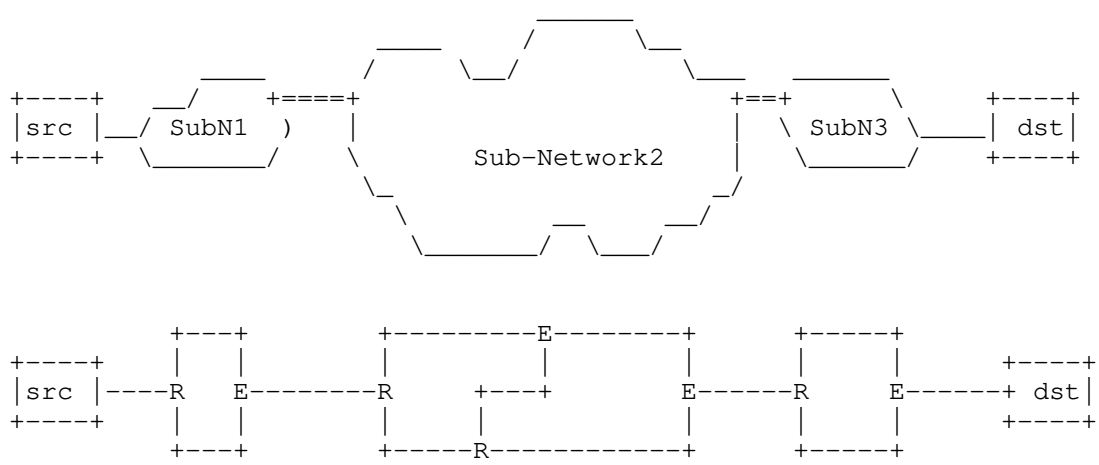


Figure 6: Replication and elimination in sub-networks for DetNet IP networks

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 IP DetNet Flows to MPLS is covered [I-D.ietf-detnet-dp-sol-mpls]. The mapping of IP DetNet Flows to IEEE 802.1 TSN is covered in Section 7.

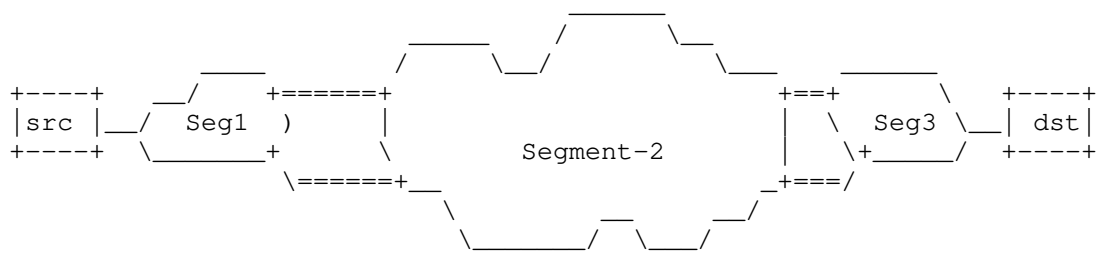


Figure 7: DetNet domains and multiple technology segments

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:

- Defend the DetNet QoS by discarding or remarking (to a non-DetNet CoS) packets received that are not the subject of a completed reservation.

- 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 transported 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 plane considerations

[Editor's note: This section needs to be different for MPLS and IP solutions. Most solutions are technology dependent.]

While management plane and control plane are traditionally considered separately, from the Data Plane perspective there is no practical difference based on the origin of flow provisioning information. This document therefore does not distinguish between information provided by a control plane protocol, e.g., RSVP-TE [RFC3209] and [RFC3473], or by a network management mechanisms, e.g., RestConf [RFC8040] and YANG [RFC7950].

[Editor's note: This section is a work in progress. discuss here what kind of enhancements are needed for DetNet and specifically for PREOF and DetNet zero congest loss and latency control. Need to cover both traffic control (queuing) and connection control (control plane).]

5.1. Explicit routes

[Editor's note: this is TBD.]

5.2. Service protection

[Editor's note: this is TBD.]

5.3. Congestion protection and latency control

[Editor's note: this is TBD.]

5.4. Flow aggregation control

[Editor's note: this is TBD.]

5.5. Bidirectional traffic

[Editor's note: This is managed at the management plane or controller level.]

Some DetNet applications generate bidirectional traffic. While the DetNet data plane must support bidirectional DetNet flows, there are no special bidirectional features with respect to the data plane other than need for the two directions take the same paths. 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. Note, that there is no stated requirement for bidirectional DetNet flows to be supported using the same 6-tuple in each direction. Control mechanisms will need to support such bidirectional flows but such mechanisms are out of scope of this document. An example control plane solution for MPLS can be found in [RFC7551].

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: using IP multicast destination address is also allowed.

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 based the next protocol values defined in Section 6.1.2. Other, non-zero values, SHOULD 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 bimask 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

[Authors note: the use of the IPv6 flow label is TBD this section requires discussion. Flow label based mapping requires src/dst adress mapping as well.]

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 fields to be ignored for a specific DetNet flow. When this fields 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.

[Authors note: Other candidate protocols include IP in IP, GRE, DCCP - should and of these be required supported?]

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:

- IPv4 and IPv6 source address field.
- IPv4 and IPv6 source address prefix length, where a zero (0) value effectively means that the address field is ignored.
- IPv4 and IPv6 destination address field.
- IPv4 and IPv6 destination address prefix length, where a zero (0) effectively means that the address field is ignored.
- 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.
- 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.
- IPv4 Type of Service and IPv6 Traffic Class Fields.

- IPv4 Type of Service and IPv6 Traffic Class Field Bitmask, where a zero (0) effectively means that theses fields are ignored.

- IPv6 flow label field. This field can be optionally used for matching. When used, can be exclusive of matching against the next header field.

- TCP and UDP Source Port. Exact and wildcard matching is required. Port ranges can optionally be used.

- TCP and UDP Destination Port. Exact and wildcard matching is required. Port ranges can optionally be used.

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 will generally be avoided in order to meet DetNet service requirements.

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 if this document MUST ensure that a DetNet flow receives the traffic treatment that is provisioned for it via management and control functions, 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 [I-D.ietf-detnet-dp-sol-mpls] and IEEE802.1 TSN Section 7. 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, bimasks, 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. Mapping IP DetNet Flows to IEEE 802.1 TSN

[Editor's note: This section is TBD - it covers how IP DetNet flows operate over an IEEE 802.1 TSN sub-network. BV to take a pass at filling in this section]

This section covers how IP DetNet flows operate over an IEEE 802.1 TSN sub-network. Figure 8 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) IP DetNet End System, (2) IP DetNet Edge or Relay node or (3) IP End System.

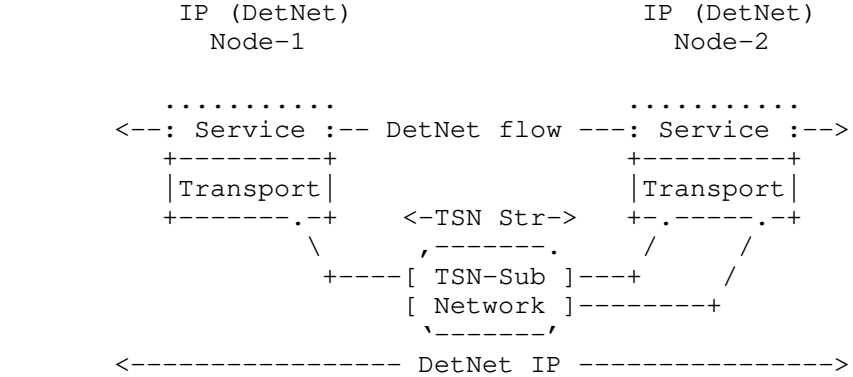


Figure 8: DetNet (DN) Enabled IP Network over a TSN sub-network

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.

7.1. TSN Stream ID Mapping

IP DetNet 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.

From the TSN sub-network perspective IP DetNet 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 IP DetNet 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 9.

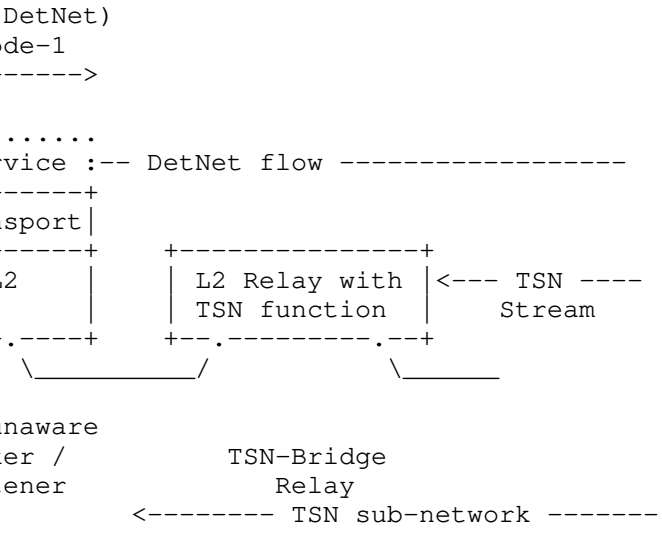


Figure 9: IP (DetNet) node without TSN functions

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 10. 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?]

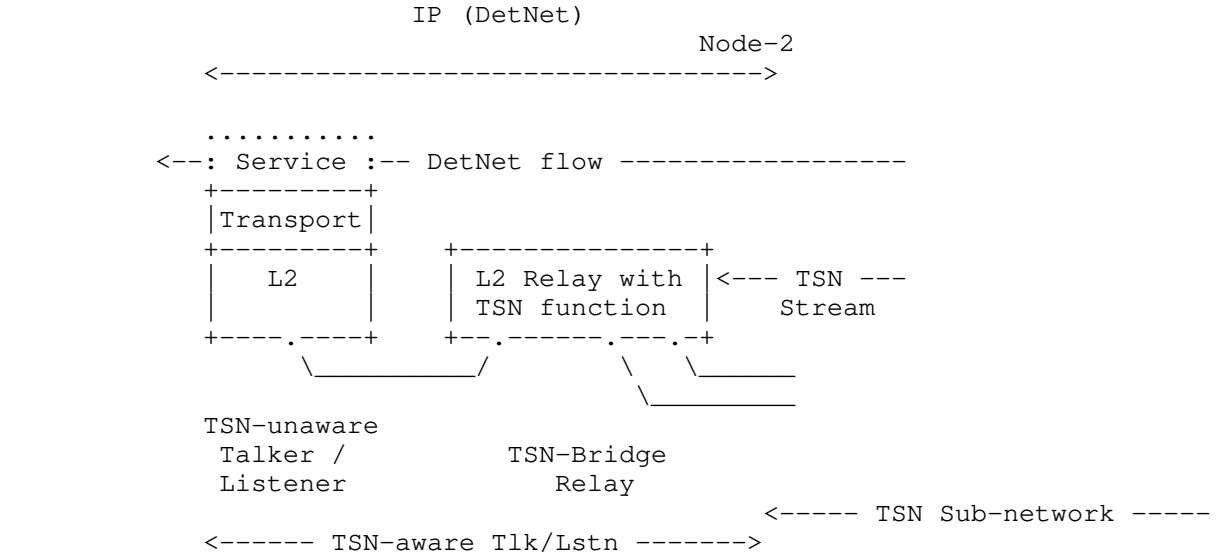


Figure 10: IP (DetNet) node with TSN functions

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].

7.2. 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.

7.3. Procedures

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

7.4. 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.]

8. 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.

9. IANA considerations

TBD.

10. 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 11.

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

11. 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

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.

12. References

12.1. 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>.

12.2. 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-08 (work in progress), September 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-flow-information-model]
Farkas, J., Varga, B., rodney.cummings@ni.com, r., Jiang, Y., and Y. Zha, "DetNet Flow Information Model", draft-ietf-detnet-flow-information-model-01 (work in progress), March 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-03 (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>.

[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>.

[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>.

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

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

Email: balazs.a.varga@ericsson.com

DetNet
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2019

J. Korhonen, Ed.

B. Varga, Ed.
Ericsson

October 21, 2018

DetNet MPLS Data Plane Encapsulation
draft-ietf-detnet-dp-sol-mpls-01

Abstract

This document specifies Deterministic Networking data plane encapsulation solutions. The described data plane solutions is applied 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 April 24, 2019.

Copyright Notice

Table of Contents

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].

This document specifies the DetNet data plane 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 layer functions and allow flow identification as described in the DetNet Architecture.

The DetNet transport layer functionality that provides congestion protection for DetNet flows is assumed to be in place in a DetNet node.

Furthermore, this document also describes how DetNet flows are identified, and how a DetNet Relay/Edge/Transit nodes works. It also describes the function and operation of the Packet Replication (PRF) Packet Elimination (PEF) and Packet Ordering (POF) functions in the MPLS data plane.

This document does not define the associated control plane functions, or Operations, Administration, and Maintenance (OAM). It also does not specify traffic handling capabilities required to deliver congestion protection and latency control for DetNet flows at the DetNet transport layer.

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 DetNet Data Plane Solution Alternatives [I-D.ietf-detnet-dp-alt].

T-Label A label used to identify the LSP used to transport 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 layer functions. An S-Label is also used to identify a DetNet flow at DetNet service layer.

PEF A Packet Elimination Function (PEF) eliminates duplicate copies of packets received by an edge or a relay node to prevent excess packets flooding the network, or to prevent duplicate packets being sent out of the DetNet domain.

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 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.

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 an edge node, a relay node, or an end system.

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

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

2.2. Abbreviations

The following abbreviations used in this document:

AC Attachment Circuit.

CE	Customer Edge equipment.
CoS	Class of Service.
CW	Control Word.
d-CW	DetNet 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. MPLS DetNet 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 layers: a service layer and a transport layer. The basic approach defined in this document supports the DetNet service layer based on existing pseudowire (PW) encapsulations and mechanisms, and supports the DetNet transport layer based on existing MPLS Traffic Engineering encapsulations and mechanisms. Background on PWs can be found in [RFC3985] and [RFC3031].

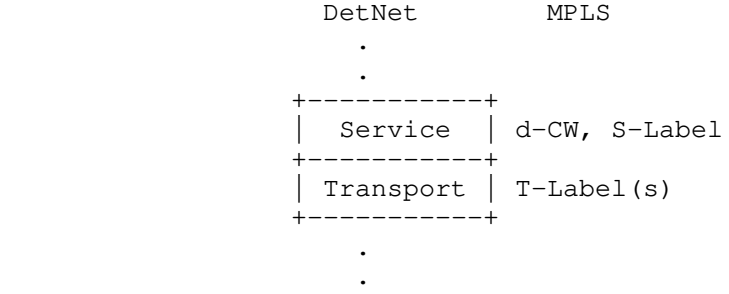


Figure 1: DetNet adaptation to MPLS data plane

The MPLS DetNet data plane approach defined in this document is shown in Figure 1. The service 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. The transport layer is supported by one or labels (T-Labels).

A node operating on a DetNet flow in the Detnet 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 to determine what local operation(s) are applied to that packet. The S-label has to be unique on each edge and relay node, which is achieved by using a label taken from the platform label space [RFC3031].

4.2. MPLS DetNet data plane scenarios

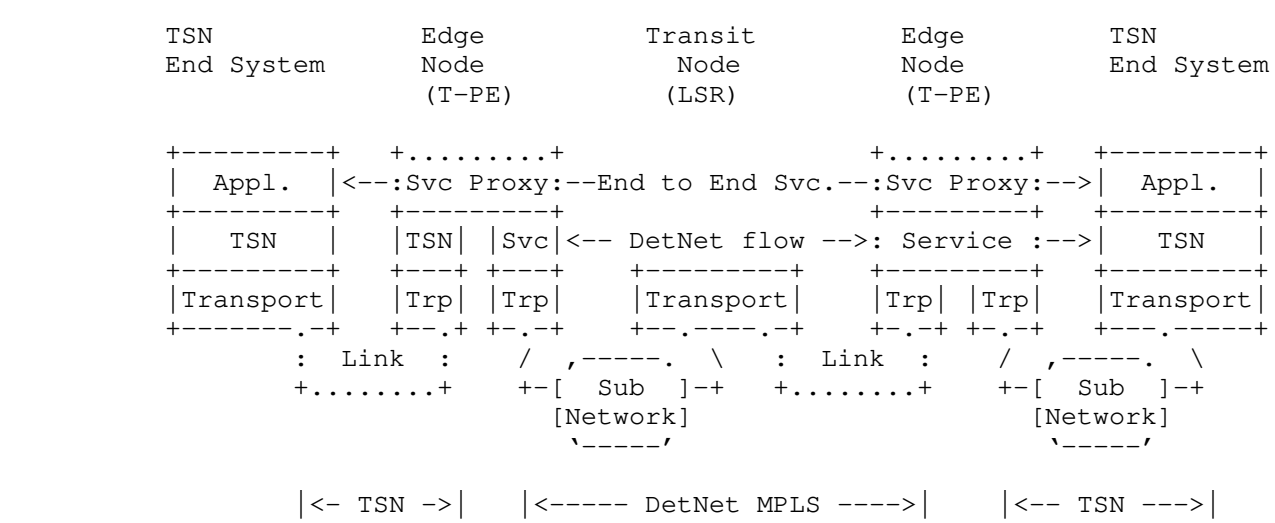


Figure 2: A TSN over DetNet MPLS Enabled Network

Figure 2 shows several node types defined in [I-D.ietf-detnet-architecture]. DetNet Edge Nodes sit at the boundary of a DetNet domain. They are responsible for mapping non-DetNet aware 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.

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. Edge Nodes MUST provide a Service Proxy entity that "associates" the DetNet flows and native flows at the edge of the DetNet domain. It ensures that the DN Flow is properly served at the Edge node (and inside the domain).

Transit nodes are normal MPLS Label Switching Routers (LSRs). They are generally unaware of the special requirements of DetNet flows, although they need to provide traffic engineering services and proper QoS to the LSPs associated with DetNet flows to enhance the prospect of the LSPs meeting the DetNet service requirements. Some implementations of transit nodes may be DetNet aware, but such nodes just support the DetNet transport layer.

The MPLS LSP may be provided by any MPLS method (provisioned, RSVP-TE, MPLS- TP, or MPLS Segment Routing (SR)).

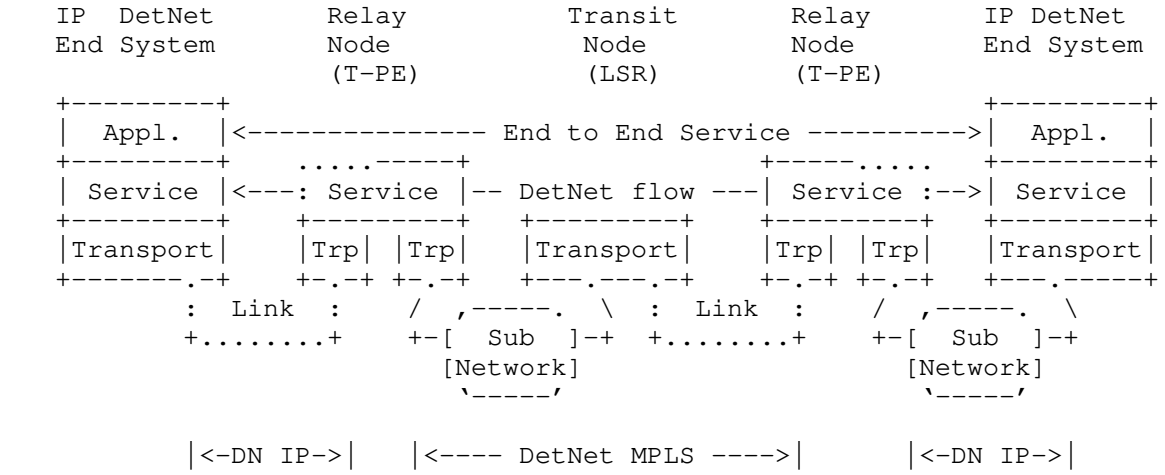


Figure 3: DetNet (DN) IP Over MPLS Network

Figure 3 and Figure 4, show different cases where relay nodes may be used. Relay nodes are similar to edge nodes in that both are aware of the needs of particular DetNet flows and take care to process them in accordance with the required performance needs. They differ in that relay nodes sit within a DetNet domain while edge nodes always sit at DetNet domain boundaries. Both node types can enhance the reliability of delivery by enabling the replication of packets so that multiple copies, possibly over multiple paths are forwarded through the DetNet domain. They also reduce the impact of replication by eliminating surplus copies of DetNet packets. Relay nodes may sit the boundary of an MPLS domain when the non-MPLS domain is DetNet aware. Relay nodes are functionally similar to PW S-PEs or, when at the edge of an MPLS network, T-PEs [RFC6073].

Figure 4 illustrates how DetNet can provide services for IEEE 802.1TSN end systems, CE1 and CE2, over a DetNet enabled network. 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 flow packet replication and elimination point. This conceptually parallels L2VPN services, and could leverage existing related solutions as discussed below.

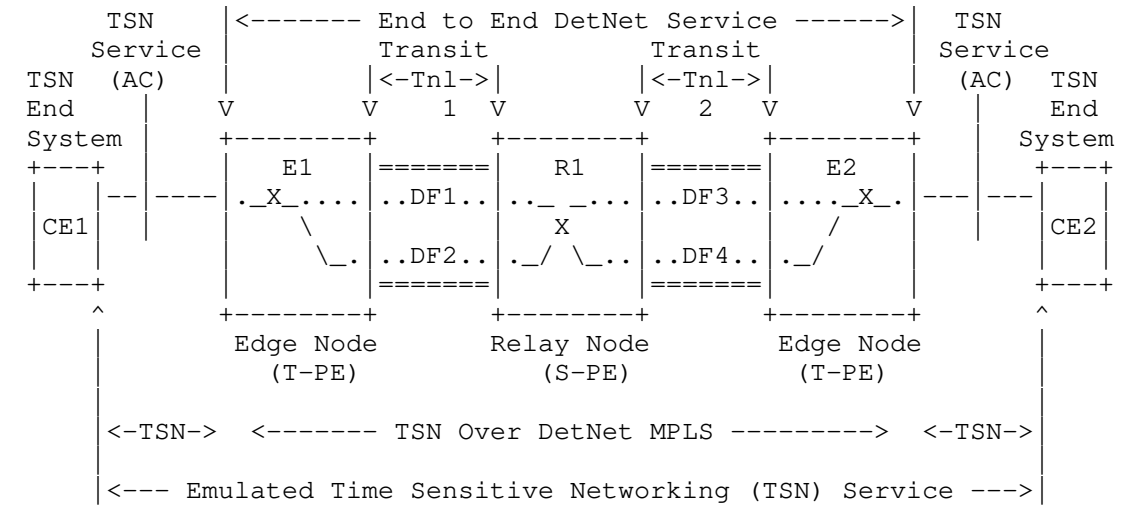


Figure 4: IEEE 802.1TSN over DetNet

[Editor's note: TSN Over DetNet MPLS arrows extended beyond the 'X' (the PREF points).]

Figure 5 illustrates how an end to end MPLS-based DetNet service is provided in a more detail. In this case, the end systems, CE1 and CE2, are able to send and receive DetNet flows, and R1 and R2 are relay nodes as they sit in the middle of a DetNet network. For example, an end system sends data encapsulated in MPLS. The 'X' in the end systems, and relay nodes represents potential DetNet flow packet replication and elimination points. In this figure, the relay nodes may change the underlying transport, for example tunneling MPLS over IP Section 11, or simply interconnect network segments.

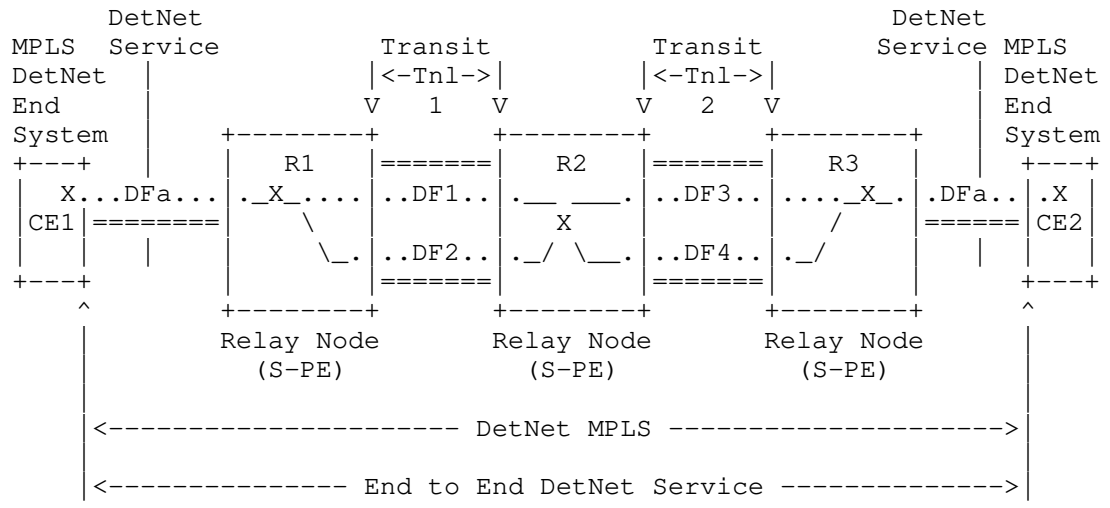


Figure 5: MPLS-Based Native DetNet

Figure 6 illustrates how an end to end MPLS-based DetNet service is provided where the end systems are not able to send and receive DetNet flows. In this example, the nodes labeled CE1 and CE2 could be non-DetNet aware IP routers or hosts. Note that E1 and E2 are edge nodes as they sit boundaries of the DetNet enabled domain.

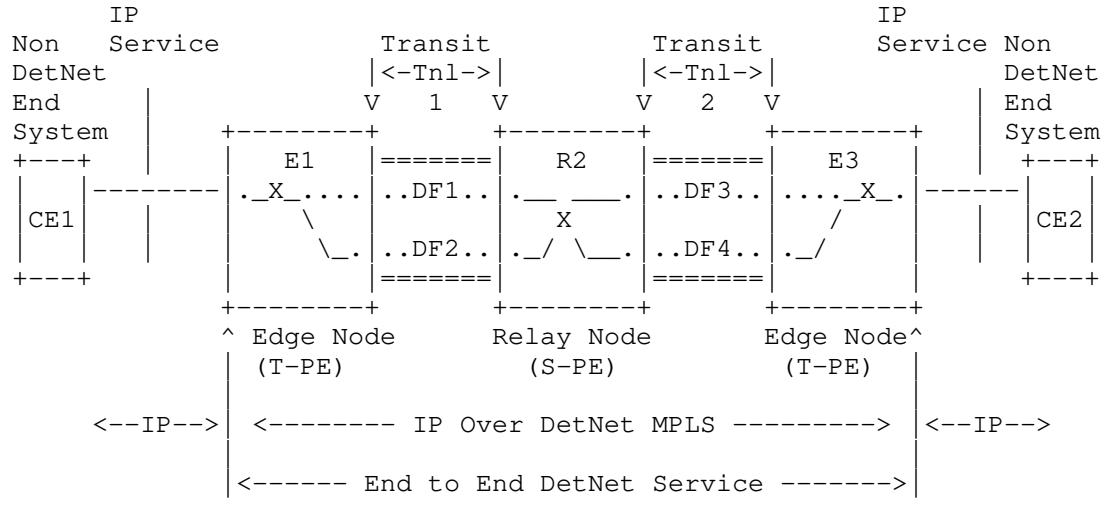


Figure 6: MPLS-Based DetNet (non-MPLS End System)

Figure 7 illustrates how end to end DetNet service is provided where the end systems are able to send and receive IP DetNet flows, e.g.,

per [I-D.ietf-detnet-dp-sol-ip], and the MPLS nodes optionally provide service protection. In this case R1 and R3 are T-PEs and R2 is an S-PE and the DetNet service is end-to-end.

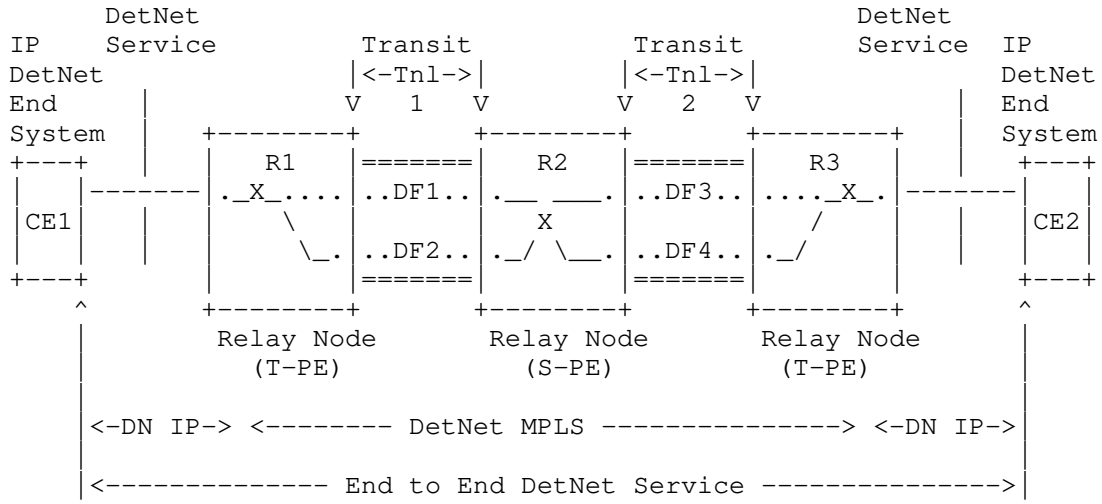


Figure 7: DetNet IP over DetNet (DN) MPLS

4.3. Packet flow example (service protection)

An example MPLS DetNet network fragment and packet flow is illustrated in Figure 8.

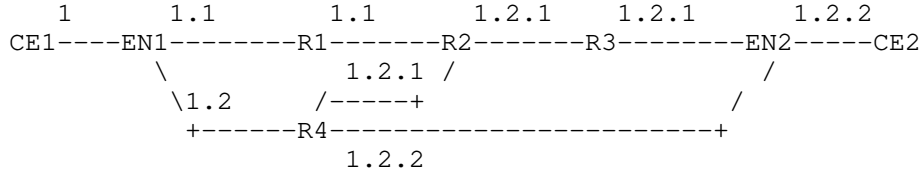


Figure 8: Example Packet flow in DetNet Enabled MPLS Network

In Figure 8 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 8.

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 transport nodes that simply forward the DetNet traffic, but these are omitted for clarity.

5. DetNet MPLS Data 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 miss-ordering and 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 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.

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 may have limited ability to provide service on per-flow DetNet identifiers. Therefore, identifying each individual DetNet flow on a transit node may not be achieved in some network scenarios, but DetNet service can still be assured in these scenarios through resource allocation and control.

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 (or even a TSN) domain the DN (TSN) functions use at most two flow parameters, namely Flow-ID and Sequence Number. However, an application may exchange further flow related parameters (e.g., time-stamp), which are not considered by DN functions.

Two types of end-systems are distinguished:

- L2 (Ethernet) end-system: application directly over L2.

- L3 (IP) end-system: application over L3.

Note: An MPLS DetNet end system (as shown in Figure 5) can be treated as a combination of an L3 (IP) end-system and an MPLS DetNet edge node.

In case of Ethernet end-systems the application data is encapsulated directly in L2. From the DN domain perspective no upper layer protocols are visible. The Data-flow uses only Ethernet tag(s) and further flow specific parameters (if needed) are hidden inside the protocol data unit (PDU).

The IP end-system scenario is different. In this case, data-flows are encapsulated directly in IP and, typically, other higher layer protocols such as UDP and Real-time Transport Protocol (RTP). Many valid combinations exist and it is up to applications to select specific headers to be used. Details on support for DetNet IP data flows can be found in [I-D.ietf-detnet-dp-sol-ip].

As a general rule, DetNet domains MUST be capable of forwarding any Data-flows and the DetNet domain MUST NOT mandate the end-system encapsulation format.

Furthermore, no application-level-proxy function is envisioned inside the DetNet domain, so end-systems peer with end-systems using the same application encapsulation format (see figure below):

- L2 end-systems peer with L2 end-systems and

- L3 end-systems peer with L3 end-systems.

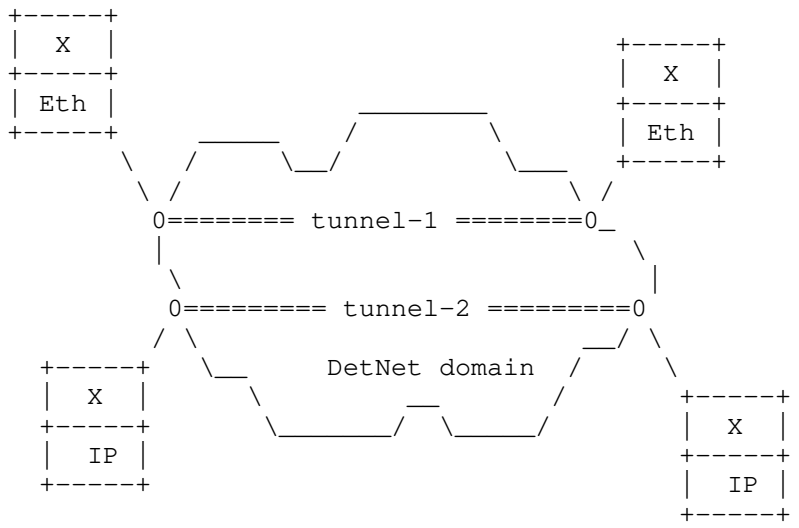


Figure 9: End-systems and the DetNet domain

5.2. DetNet domain specific considerations

From a connection type perspective, three scenarios are distinguished:

1. Directly attached: end-system is directly connected to an edge node.

2. Indirectly attached: end-system is behind a (L2-TSN / L3-DetNet) sub-network.

3. DN integrated: end-system is part of the DetNet domain.

L3 end-systems may use any of these connection types, however L2 end-systems may use only the first two (directly or indirectly attached). DetNet domain MUST allow communication between any end-systems of the same type (L2-L2, L3-L3), independent of their connection type and DetNet capability. However directly attached and indirectly attached end-systems have no knowledge about the DetNet domain and its encapsulation format at all. See Figure 10 for L3 end-system scenarios.

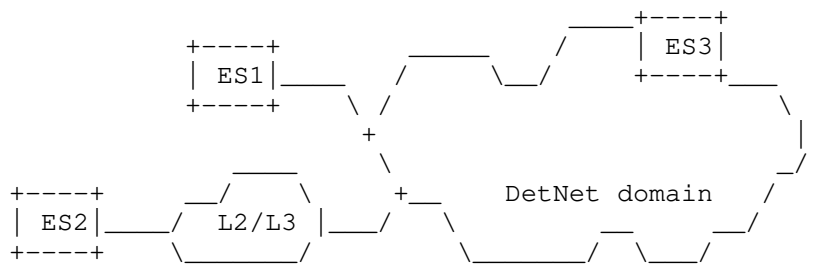
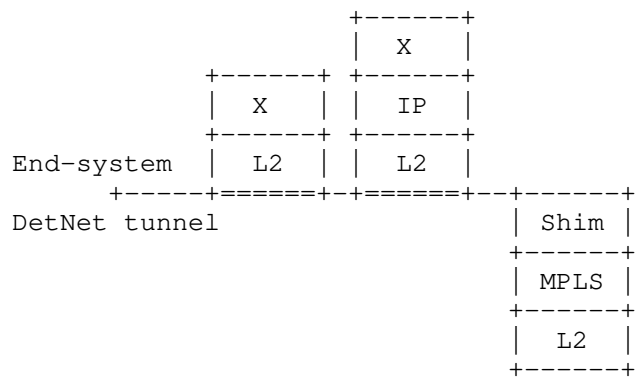


Figure 10: Connection types of L3 end-systems

5.2.1. DetNet Layer Two Service

The simplest DetNet service is to provide tunneling for layer two, 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), or on received interface [RFC3985]. In both cases the L2 headers MUST either be kept, or provision must be made for their reconstruction at egress from the DetNet domain.



Examples:

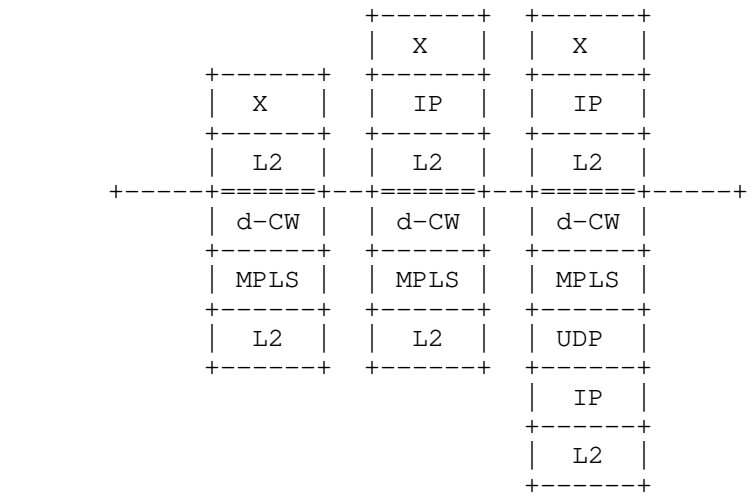


Figure 11: Encapsulation format for DetNet Layer Two Service

As shown in Figure 11 both L2 and L3 end-systems can be served by such a DetNet L2 encapsulation service. This encapsulation service may be carried over MPLS natively Section 6.2, or over MPLS over IP Section 11.

5.2.2. DetNet Routing Service (IP over MPLS)

IP traffic and IP DetNet flows, see [I-D.ietf-detnet-dp-sol-ip], can be carried over a DetNet MPLS domain. In such cases, the IP headers are modified per standard router behavior, e.g., TTL handling.

Figure 12 shows the encapsulation of an IP flow over MPLS as well as when MPLS is carried over an IP PSN, see Section 11.

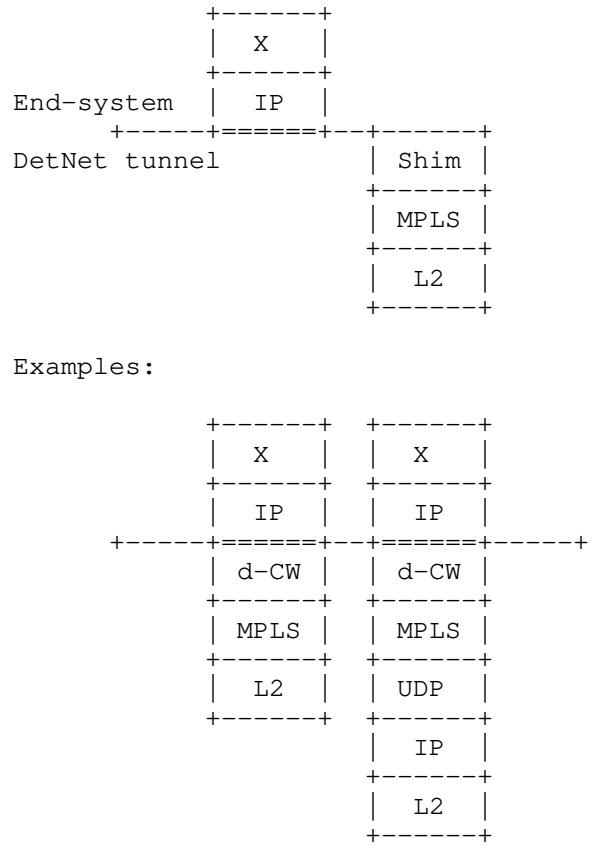


Figure 12: Encapsulation format for DetNet Routing in MPLS PSN for L3 end-systems

5.3. DetNet Inter-Working Function (DN-IWF)

5.3.1. Networks with multiple technology segments

There are networking scenarios, where the DetNet domain contains multiple technology segments (IP, MPLS, ..) and all those segments are under the same administrative control (see Figure 13). Furthermore, DetNet nodes may be interconnected via TSN segments.

An important aspect of DetNet network design is the placement of DetNet functions across the domain. Designs based on segment-by-segment optimization can provide only sub-optimal solutions. In order to achieve global optimized Inter-Working Functions (DN-IWF) can be placed at segment edge nodes, which stitch together DetNet flows across connected segments.

DN-IWF may ensure that flow attributes are correlated across segment edges. For example, there are two DetNet functions which require Sequence Numbers: (1) PEF: removes duplications from flows and (2) POF: ensures in-order-delivery of packet in a flow. Stitching flows together and correlating attributes means for example that replication of packets can happen in one segment and elimination of duplicates in a different one.

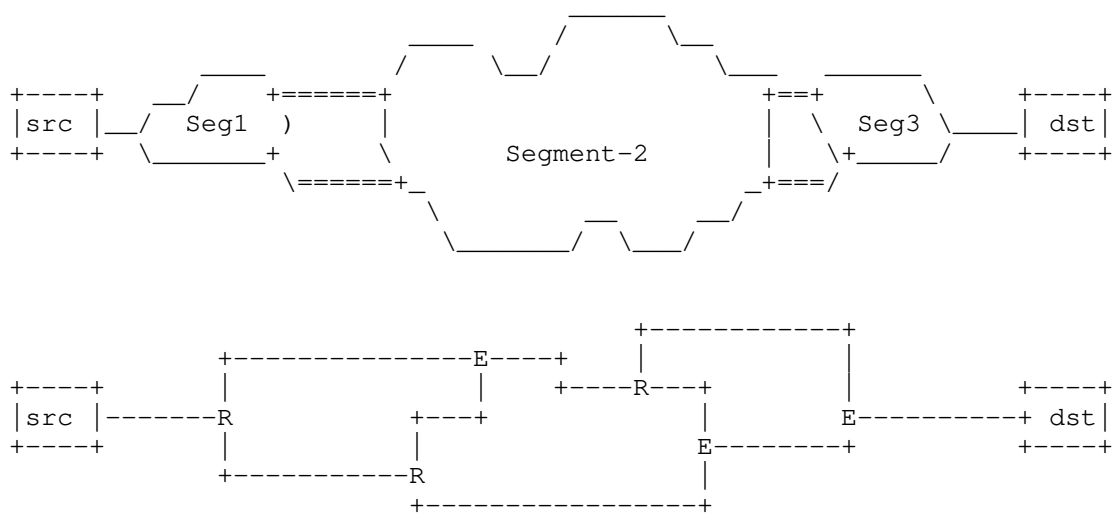


Figure 13: Optimal replication and elimination placement across technology segments example

5.3.2. DN-IWF related considerations

The goal of DN-IWF is to (1) match and (2) translate segment specific flow attributes. The DN-IWF ensures that segment specific attributes comprise per domain unique attributes for the whole DetNet domain. This characteristic can ensure that DetNet functions can be based on per domain attributes and not per segment attributes.

The two DetNet specific attributes have the following characteristics:

- Flow-ID: it is same in all packets of a flow

- Sequence Number: it is different packet-by-packet

For the Flow-ID the DN-IWF can implement a static mapping. The situation is more complicated for Sequence Number as it is different packet-by-packet, so it may need more sophisticated translation unless its format is exactly the same in the two technology segments. In this later case the DN-IWF can simple copy the Sequence Number field between the tunneling encapsulation of the two technology segments.

In case of three technology segments (IP, MPLS and TSN) three DN-IWF functions can be specified. In the rest of this section the focus is on the (1) IP - MPLS network scenario. Note: the use-cases are out-of-scope for (2) TSN - IP, (3) TSN - MPLS.

Simplest implementation of DN-IWF is provided if the flow attributes have the same format. Such a common denominator of the tunnel encapsulation format is the pseudowire encapsulation over both IP and MPLS.

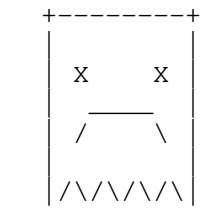


Figure 14: FIGURE Placeholder PW over X

[Editor's note: Where is the text describing how 802.1 TSN Streams are mapping to DetNet services/flows. i.e., EVPN+]

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 sequence number is carried in the DetNet Control word which carries the Data/OAM discriminator. The LSP used to transport 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 (T-Label) label and/or the S-Label may be used to indicate the queue processing as well as the forwarding parameters.

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. In addition it must be possible to send a packet with a zero length sequence number, to support the case where sequence numbers are not required by a particular DetNet flow.

Note that the concept of a zero length sequence number is not to be confused with a sequence number of zero. For example, were the sequence number size is 16 bits, the sequence will contain: 65535, 0, 1. In this case zero is an ordinary sequence number. Unlike [RFC4448] a sequence number of zero does not indicate that no sequence number is in use. Where sequence numbers are not in use, and thus a zero length sequence number is in used, the sequence number field in the packet is sent as zero. The DetNet packet forwarder knows which of these cases applies through configuration parameters associated with each specific DetNet flow.

Note that when the network consists only of DetNet enabled nodes with no aggregation, Penultimate Hop Popping (PHP) means that the only label in the label stack may be the S-label.

6.2. MPLS data plane encapsulation

Figure 15 illustrates a DetNet data plane MPLS encapsulation. The MPLS-based encapsulation of the DetNet flows is a good fit for the Layer-2 interconnect deployment cases (see Figure 4). Furthermore, end to end DetNet service i.e., native DetNet deployment (see Figure 5) 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:

- DetNet control word (d-CW) containing sequencing information for packet replication and duplicate elimination purposes, and the OAM indicator. There MUST be a separate sequence number space for each DetNet flow.

- DetNet service Label (S-label) that identifies a DetNet flow to the peer node that is to process it. The S-Label is allocated from the platform label space [RFC3031].

- Zero or more MPLS transport LSP label(s) (T-label) used to direct the packet along the label switched path (LSP) to the next peer node along the path. When Penultimate Hop Popping is in use there may be no label T-label in the protocol stack on the final hop.

- The necessary data-link encapsulation is then applied prior to transmission over the physical media.

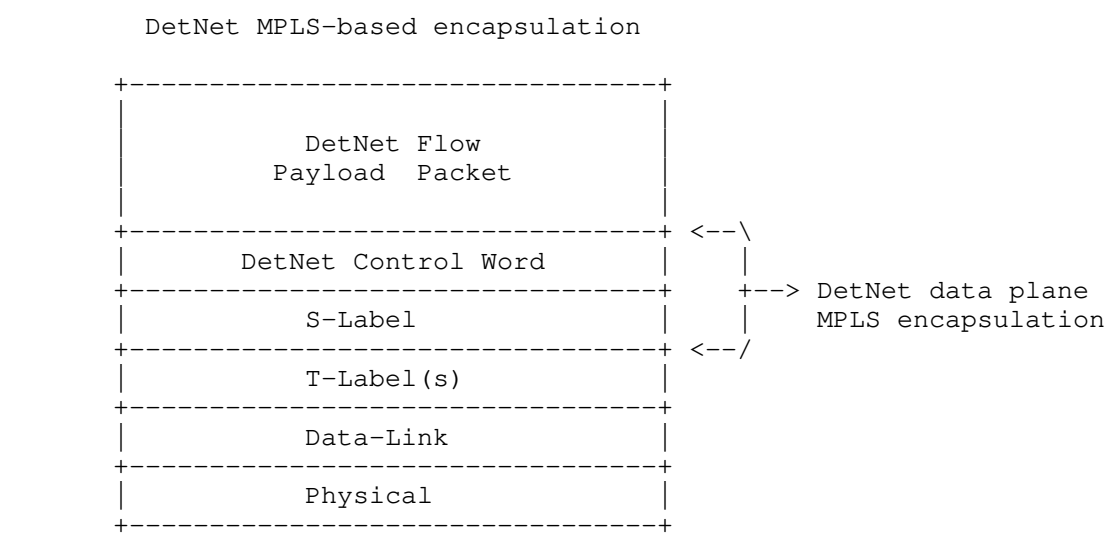


Figure 15: Encapsulation of a DetNet flow in an MPLS(-TP) PSN

6.3. DetNet control word

A DetNet control word (d-CW) conforms to the Generic PW MPLS Control Word (PWMCW) defined in [RFC4385] and is illustrated in Figure 16. The upper nibble of the d-CW MUST be set to zero (0). Two sequence number sizes are supported: 16 bits and 28 bits. The sequence number size in use for the d-CW associated with a DetNet flow (S-Label) is configured either by a control plane or manually for each DetNet flow. The sequence number is aligned to the right (least significant bits) and unused bits MUST be set to zero (0). Each DetNet flow MUST have its own sequence number counter. The sequence number is incremented by one for each new packet.

As discussed in Section 6, zero is an ordinary sequence number with no special meaning. Also as discussed therein, where no sequence number is used by a particular DetNet flow, the sequence number field in the d-CW is set to zero.

The d-CW MUST always be present in a packet. In a case where the sequence number is not used (e.g., for DetNet-t-flows) a zero length sequence number is used and the sequence number MUST be set to zero (0).

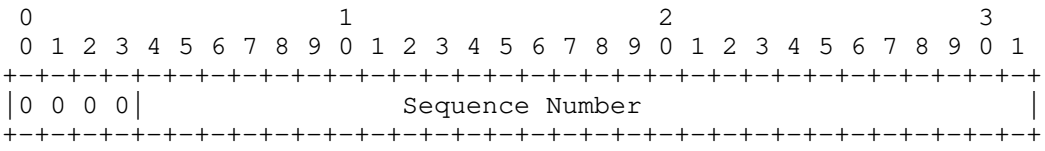


Figure 16: DetNet Control Word

6.4. Flow Identification

DetNet flow identification at a DetNet service layer is realized by an S-label. The S-label is allocated from the platform label space [RFC3031] which means that the DetNet flow is correctly identified and matched to the flow parameters, including the flow history, regardless of which input interface the packet arrives on. The S-label MUST be at the bottom label of the label stack for a DetNet-s- or DetNet-st-flow and MUST precede the d-CW.

The S-label for a specific DetNet flow is unique to that DetNet flow on a specific node, but is not required to be identical with the S-label for that DetNet flow in any other node within the DetNet domain. Thus the S-label can only be used to identify the DetNet flow at the intended receiving node.

6.5. Indication of the DetNet Payload Type

The only nodes that needs to know the payload type of a flow are the DetNet ingress node and the DetNet egress nodes. The ingress node has to know how to process the packet it receives from the ingress AC or IP flow, and the egress edge node has to know how to prepare the packet for transmission to the next hop.

On ingress a DetNet edge node has to classify the packets into those that are for transmission as Detnet packets and those that are for transmission as "normal" packets at one of more lower priorities. The packet type is indicated to the egress edge node through the value of the S-label. Thus, when the egress edge node looks up the S-label one of the parameters returned is the packet type which in turn tells the egress edge node how to prepare the packet for transmission to a next hop.

The consequence of this approach is that if multiple packet encapsulations are processed on a node pair, each encapsulation will need its own S-Label. That is not generally a problems, since it is anticipated that only one encapsulation type will be present for each DetNet flow. Of course, if for some reason the multiple encapsulations are needed to support a single DetNet service,

multiple S-labels will be required for that service. Note that in the unlikely case that Ipv4 and IPv6 will map to the same DetNet flow, different S-labels will be needed to differentiate between the versions of IP.

6.6. 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.

6.7. Flow Aggregation

1. Aggregate at the LSP (Transport)

2. Aggregating DetNet flows as a new DetNet flow

3. Simple Aggregation at the DetNet layer

A further method of using SR to perform aggregation is for further study.

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:

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 T-label.

End of review comment.

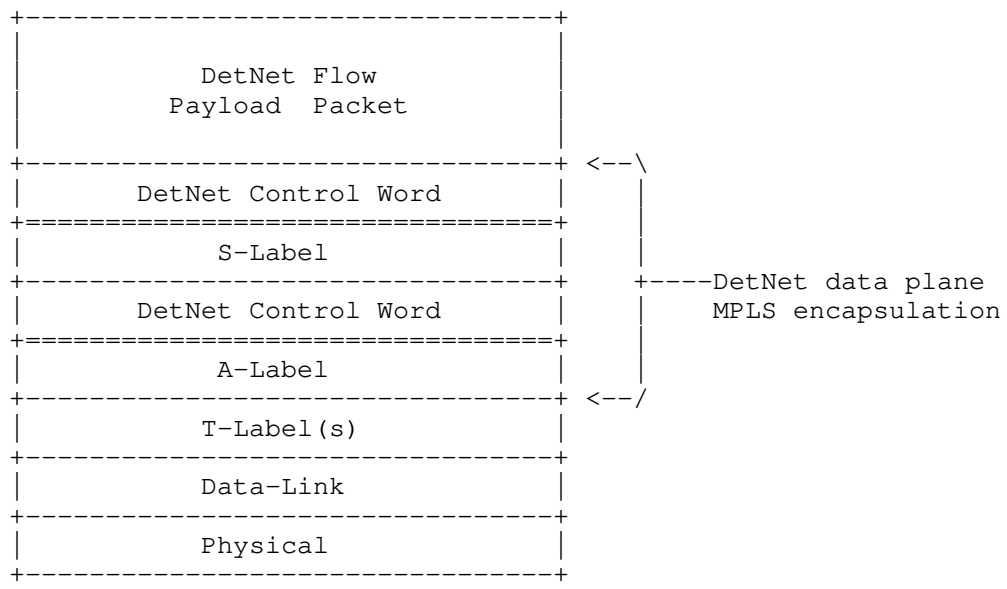
6.7.1. Aggregation at the LSP

DetNet flows transported 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.

6.7.2. Aggregating DetNet flows as a new DetNet flow

An aggregate can be built by layering DetNet flows as shown below:

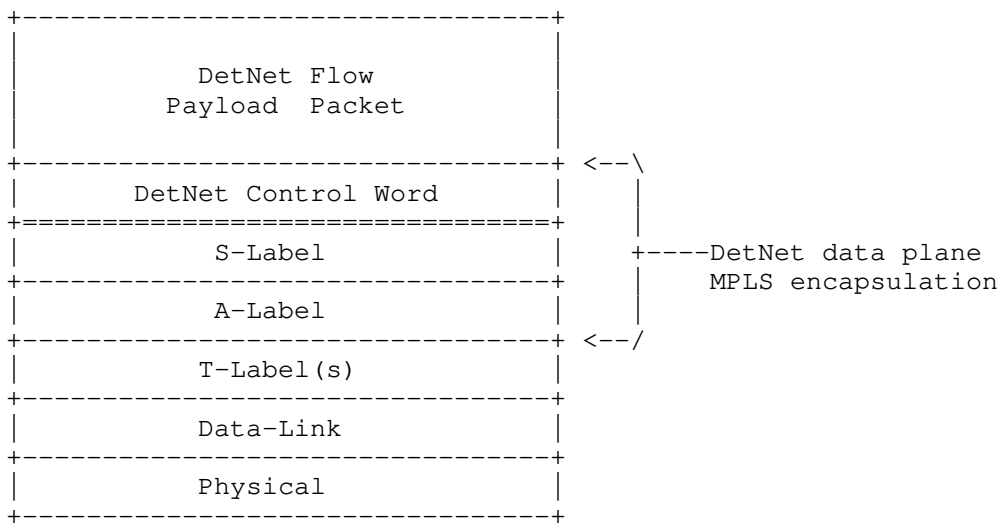


Both the Aggregation (A) label and the S-label have their MPLS S bit set indicating bottom of stack, and the d-CW allows the PREOF to work.

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.

6.7.3. 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 transport 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.



6.8. Service 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.

6.8.1. Edge node processing

An edge node is resposable 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 is such as 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 MPLS DetNet" 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).

6.8.2. Relay node processing

A DetNet Relay node operates in the DetNet transport 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 6.9.3. Again, this is entirely up to how the forwarding has been programmed.

6.9. Other DetNet data plane considerations

6.9.1. Class of Service

[Editor's note: this section needs to updated to discuss how DetNet service is mapped to E- and L-LSPs. Perhaps this gets merged with the aggregation section or dropped?]

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].

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].

6.9.2. 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, 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. 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.

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 defend the DetNet QoS by discarding or remarking (to a non-DetNet CoS) packets received that are not the subject of a completed reservation.

- MUST 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.

6.9.3. Cross-DetNet flow resource aggregation

[Editor's NOTE: keep and extend this section.]

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 transported 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.

DetNet flows transported 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.

6.9.4. Layer 2 addressing and QoS Considerations

[Editor's NOTE: review and simplify this section.]

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 IPv6 encapsulations.

6.9.5. 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. Management and control considerations

[Editor's note: This section needs to be different for MPLS and IP solutions. Most solutions are technology dependant. Currently most text in this section is just a draft and may have bits that are already moved to other places/documents.]

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. This document therefore does not distinguish between information provided by a control plane protocol, e.g., RSVP-TE [RFC3209] and [RFC3473], or by a network management mechanisms, e.g., RestConf [RFC8040] and YANG [RFC7950].

[Editor's note: This section is a work in progress. discuss here what kind of enhancements are needed for DetNet and specifically for PREOF and DetNet zero congest loss and latency control. Need to cover both traffic control (queuing) and connection control (control plane).]

7.1. MPLS-based data plane

7.1.1. S-Label assignment and distribution

[Editor's note: Outdated and needs more work.]

The DetNet S-Label distribution follows the same mechanisms specified for XYZ . The details of the control plane protocol solution required for the label distribution and the management of the label number space are out of scope of this document.

7.1.2. Explicit routes

It is necessary to consider explicit routes both at the DetNet layer and in the MPLS layer. In the DetNet layer the explicit route consists of the set of Relay Nodes that the DetNet flow must

traverse. In the MPLS layer the explicit route consists of the set of LSRs, links, and possibly link bundle members and queues that the DetNet packets of a flow must traverse between nodes in the DetNet 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. This detailed steering is needed to ensure that packets are routed through the resources that have been reserved for them, and hence provide the DetNet application with the required performance.

Whether configuring, calculating and instantiating this is a multi-stage process, or a single stage process is out of scope of this document.

The one method of explicitly setting up the explicit path at the DetNet layer is through the use of the management controller.

[Editor's note: a method of setting up a graph through the DetNet Nodes using the IGP has been proposed. A reference is needed to e.g., RFC 7813 IS-IS Path Control and Reservation.]

There are a number of approaches that can be taken to provide explicit routes/paths in the MPLS layer:

- The path can be explicitly set up by the management controller calculating the path and explicitly configuring each node along that path.

- The LSP can be set up using RSVP-TE. Such an approach confines the packet to the explicit path.

- The path can be implemented using segment routing.

Where the DetNet traffic is carried over IP Section 11 explicit paths may need to be provided in the IP layer. This is for further study.

7.2. Packet replication and elimination

[Editor's note: Outdated and at the functional level technology independent.. but needs more work.]

The control plane protocol solution required for managing the PREOF processing is outside the scope of this document.

7.3. Congestion protection and latency control

[Editor's note: TBD]

7.4. Bidirectional traffic

[Editor's NOTE: this section needs to be updated to have its scope limited to management and control.]

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 transport path. This would be analogous of standard IP routing, or PWs running over two reciprocal unidirection 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 allocations may differ in each direction. The concepts of associated bidirectional flows and co-routed bidirectional flows can be applied to DetNet flows as well whether IPv6 or MPLS is used.

While the IPv6 and MPLS data planes must support bidirectional DetNet flows, there are no special bidirectional features with respect to the data plane other than need for the two directions take the same paths. 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 IPv6 Flow Labels or MPLS Labels in each direction. Control mechanisms will need to support such bidirectional flows for both IPv6 and MPLS, but such mechanisms are out of scope of this document. An example control plane solution for MPLS can be found in [RFC7551].

7.5. Flow aggregation control

[TBD]

8. DetNet IP Operation over DetNet MPLS Service

[Editor's note: this is a place holder section. A standalone section on operation of IP flows over DetNet MPLS data plane. Includes RFC2119 Language.]

9. IEEE 802.1 TSN Interconnection over DetNet MPLS Service

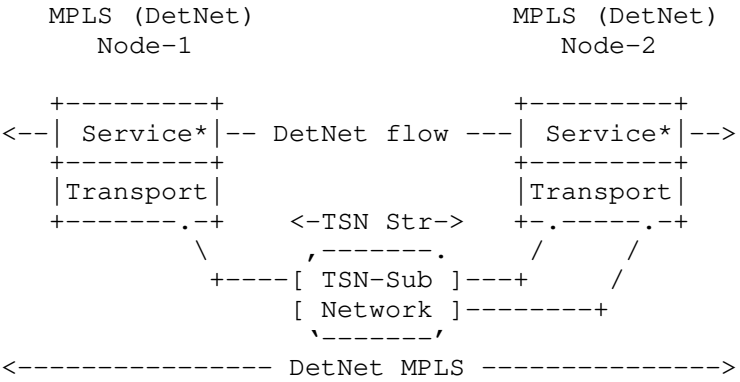
[Editor's note: this is a place holder section. A standalone section on TSN "island" interconnect over DetNet". Includes RFC2119 Language.]

10. DetNet MPLS Transport Layer 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 MPLS DetNet flows operate over an IEEE 802.1 TSN sub-network. Figure 17 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) MPLS DetNet End System, (2) MPLS DetNet Edge or Relay node or (3) MPLS Transit node.

Note: in case of MPLS Transit node there is no DetNet Service sub-layer.



Note: * no service sub-layer for transit nodes

Figure 17: 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 challange 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 MPLS DetNet nodes MUST ensure proper TSN sub-network specific Ethernet encapsulation of the MPLS DetNet 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.

10.1. Mapping of TSN Stream ID and Sequence Number

TSN capable MPLS (DetNet) nodes are TSN-aware Talker/Listener as shown in Figure 18. 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, additonaly:

    * 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 MPLS DetNet 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.

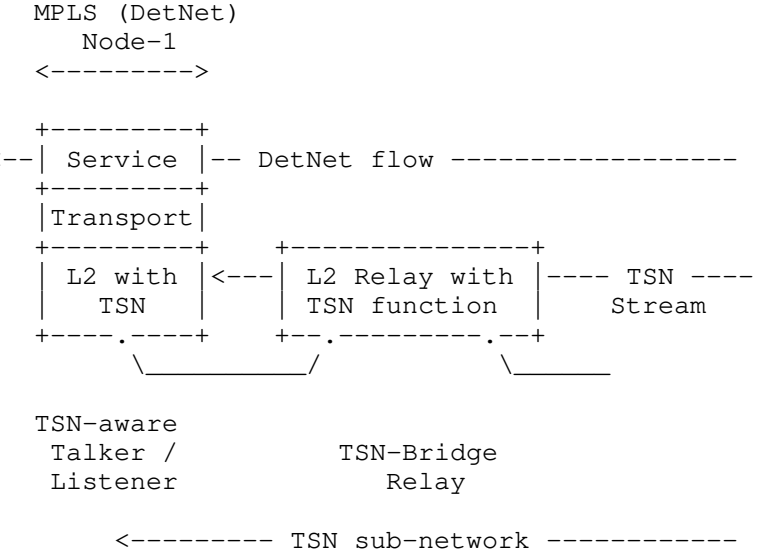


Figure 18: MPLS (DetNet) node with TSN functions

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

10.2. 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.

10.3. 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.]

11. DetNet MPLS Transport Layer Operation over IP DetNet PSNs

This section specifies the DetNet encapsulation over an IP transport network. The approach is modeled on the operation of MPLS and PseudoWires (PW) over an IP Packet Switched Network (PSN) [RFC3985][RFC4385][RFC7510]. It is also based on the MPLS data plane encapsulation described in Section 6.2.

To carry DetNet with full functionality at the DetNet layer over an IP transport 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.

To simplify operations and implementations, rather than inventing a new encapsulation, the IP encapsulation takes advantage of the MPLS

encapsulation. By using the specification of MPLS over UDP and IP in [RFC7510], the T-Label(s) shown in Figure 15 in Section 6.2 can be replaced by UDP and IP, resulting in the following encapsulation:

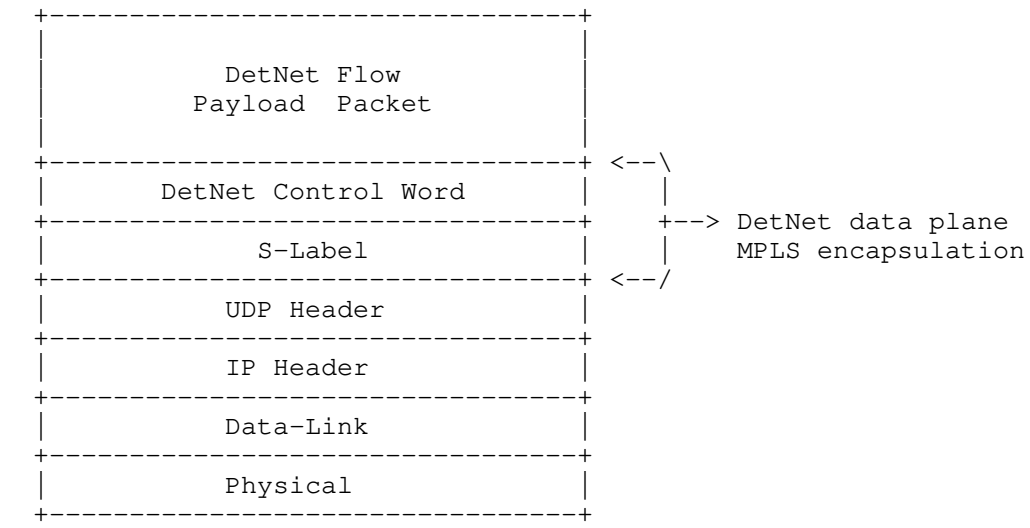


Figure 19: IP Encapsulation of DetNet

Where the UDP header is used as defined in Section 3 of [RFC7510].

As in Section 6.2, the S-Label is used to identify a DetNet flow to the peer node that processes it, in this case the node addressed by the IP Header in Figure 19. The S-Label is allocated from the receiving node?s platform label space [RFC3031].

In ingress Edge Nodes, the encapsulation in Figure 19 will be imposed on Detnet Flow Payload Packets as received from DetNet End Systems, and the encapsulation will be removed in egress Edge Nodes as they transmit the Payload Packets to the End Systems.

Note that this encapsulation works equally well with IPv4 and IPv6.

This encapsulation can also be used in conjunction with segment routing as specified in [I-D.ietf-spring-segment-routing-mpls]. In this case, the T-Label(s) in Figure 19 should be retained, and at each hop, the top T-label is popped and mapped to a corresponding UDP/IP tunnel, resulting in the following encapsulation:

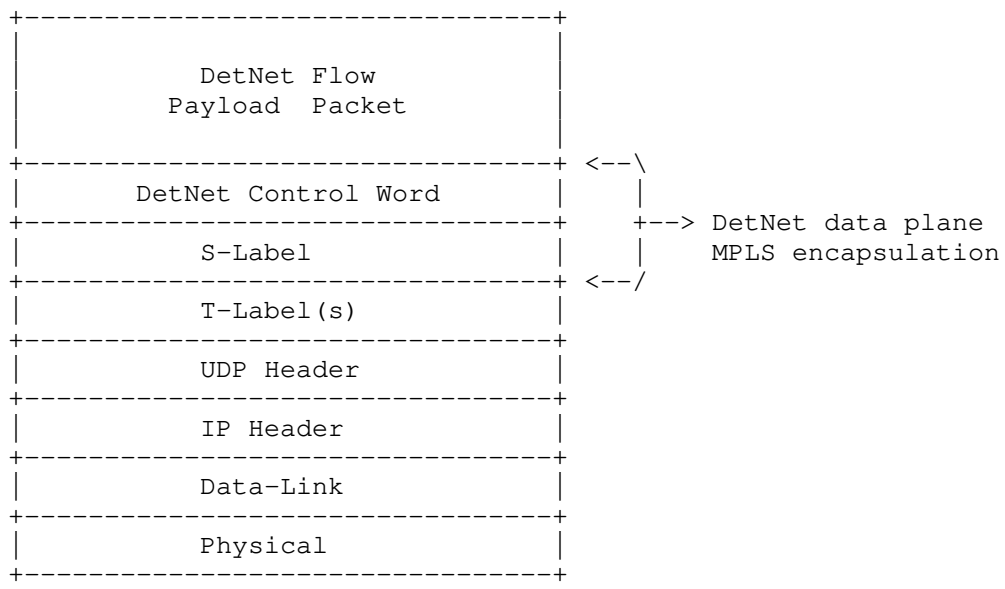


Figure 20: IP Encapsulation of DetNet with MPLS-SR

Again, the UDP header is used as defined in Section 3 of [RFC7510].

Note that if required in both the case of IP Encapsulation of DetNet Figure 19, and of IP Encapsulation of DetNet with MPLS-SR Figure 20, it is possible to omit the UDP header if required. Operation of MPLS directly over IP is described in [RFC4023]. In this case DetNet Service can be provided on a per IP flow basis as described in [I-D.ietf-detnet-dp-sol-ip].

12. 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.

13. IANA considerations

This document makes no IANA requests.

14. 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 15.

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

Stewart Bryant
Huawei Technologies
Email: stewart.bryant@gmail.com

Mach Chen
Huawei Technologies
Email: mach.chen@huawei.com

15. 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

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.

16. References

16.1. Normative references

[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-14 (work in progress), June 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>.

[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>.

[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>.

[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>.

[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>.

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, Ed., "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, DOI 10.17487/RFC4023, March 2005, <https://www.rfc-editor.org/info/rfc4023>.

[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>.

[RFC6003] Papadimitriou, D., "Ethernet Traffic Parameters", RFC 6003, DOI 10.17487/RFC6003, October 2010, <https://www.rfc-editor.org/info/rfc6003>.

[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>.

16.2. 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-08 (work in progress), September 2018.

[I-D.ietf-detnet-dp-alt]
Korhonen, J., Farkas, J., Mirsky, G., Thubert, P., Zhuangyan, Z., and L. Berger, "DetNet Data Plane Protocol and Solution Alternatives", draft-ietf-detnet-dp-alt-00 (work in progress), October 2016.

[I-D.ietf-detnet-dp-sol-ip]
Korhonen, J., Varga, B., "DetNet IP Data Plane Encapsulation", 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>.

[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>.

[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>.

[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>.

[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>.

[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

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

Email: balazs.a.varga@ericsson.com

DetNet
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

J. Farkas
B. Varga
Ericsson
R. Cummings
National Instruments
Y. Jiang
Y. Zha
Huawei Technologies Co., Ltd.
October 22, 2018

DetNet Flow Information Model
draft-ietf-detnet-flow-information-model-02

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 April 25, 2019.

Table of Contents

1. 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 distinguished:

- 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).

- 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.

- 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.

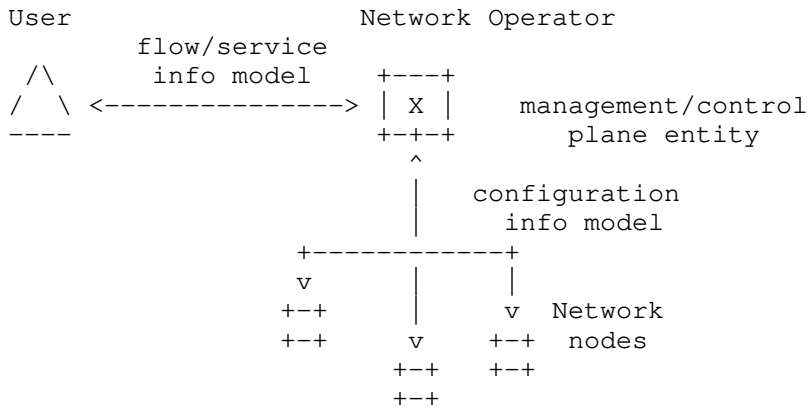


Figure 1: Usage of Information models (flow, service and configuration)

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.

1.1. 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.

1.2. 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.

2. 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.

3. 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. Additional terms used in this document:

DetNet L3 Flow: Layer 3 (L3) flow leveraging DetNet service.

DetNet L2 Flow: Layer 2 (L2) flow leveraging DetNet service.

DetNetwork Flow: DetNet data plane specific encapsulated format of a DetNet L2 or L3 flow leveraging DetNet service.

4. 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.

- Names SHOULD be descriptive.

- Names MUST start with uppercase letters.

- 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.

5. Service model

5.1. 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.

5.2. 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:

- Bandwidth
  It is the bandwidth guaranteed for the DetNet service.

- Delay parameters
  The 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.)

- 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.

- 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.

- 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.)

- 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.

5.3. 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.

Note: Further discussion is needed based on data plane encapsulation results what reference points should be defined. Only some possible examples listed here:

- App-flow endpoint: End system's internal reference point for the native data flow.

- DetNet-UNI: UNI interface ("U") on a DetNet edge node.

- 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 adds 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.]]

5.4. Service scenarios

Using the above defined reference points, two major service scenarios can be identified:

- 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).

- 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.]]

6. 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:

- 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.

- 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.

- 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:

- 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.

- 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.

- 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.

- 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 two operations for each flow with respect to a Source or a Destination (and an Ingress or an Egress):

- 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 7.2) 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 flows, end systems may not need to implement the L3 <=> L2 Transfer Function specified by [IEEE8021CB] (see, e.g., subclause 6.3; see also subclause 46.1 in [IEEE8021Qcc]). An 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].

7. Flow

The flows leveraging DetNet service can be unicast or multicast data flows for an application with constrained requirements on network performance, e.g., low packet loss rate and/or latency. Therefore, they 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 7.1)

b. TrafficSpecification (Section 7.2)

c. FlowRank (Section 7.3)

Flow attributes are described in the following sections.

7.1. Identification and Specification of Flows

Identification options for DetNet flows at the UNI and within the DetNet domain are specified as follows; see Section 7.1.1 for DetNet L3 flows (at UNI), Section 7.1.2 for DetNet L2 flows (at UNI) and Section 7.1.3 for DetNetwork flows (within the network).

7.1.1. DetNet L3 Flow Identification and Specification at UNI

DetNet L3 flows can be identified and specified by the following attributes:

a. SourceIpAddress

b. DestinationIpAddress

c. IPv6FlowLabel

d. Dscp

e. Protocol

f. SourcePort

g. DestinationPort

7.1.2. DetNet L2 Flow Identification and Specification at UNI

DetNet L2 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:

- A 48-bit MAC Address associated with the Talker sourcing the stream to the bridged network.
- A 16-bit unsigned integer value, Unique ID, used to distinguish among multiple streams sourced by the same Talker.

7.1.3. DetNetwork 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. (Protocol)

e. (SourcePort)

f. (DestinationPort)

g. MplsLabel

[[Note: attributes in brackets are dependant on current dataplane discussions.]]

7.2. 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 massage 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.]]

7.3. 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).

7.4. 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.]]

8. Source

The Source object specifies:

- The behavior of the Source for the flow (how/when the Source transmits).
- The requirements of the Source from the network.
- The capabilities of the interface(s) of the Source.

The Source object includes the following attributes:

a. DataFlowSpecification (Section 7.1)

b. TrafficSpecification (Section 7.2)

c. FlowRank (Section 7.3)

d. EndSystemInterfaces (Section 10.1)

e. InterfaceCapabilities (Section 10.2)

f. UserToNetworkRequirements (Section 10.3)

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.

9. Destination

The Destination object includes the following attributes:

a. DataFlowSpecification (Section 7.1)

b. EndSystemInterfaces (Section 10.1)

c. InterfaceCapabilities (Section 10.2)

d. UserToNetworkRequirements (Section 10.3)

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): Should we add DestinationRank? It could distinguish the importance of Destinations if the flow cannot be provided for all Destinations.]]

10. Common Attributes of Source and Destination

Source and Destination end systems have the following common attributes in addition to DataFlowSpecification (Section 7.1).

10.1. 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.

10.2. Interface Capabilities

InterfaceCapabilities specifies the network capabilities of all interfaces (ports) contained in the EndSystemInterfaces object (Section 10.1). These capabilities may be configured via the InterfaceConfiguration object (Section 14.2) of the Status object (Section 14).

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 10.1)). 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)

[[NOTE (to be removed from a future revision): InterfaceCapabilities attributes are to be defined. For information, [IEEE8021Qcc] specifies the following attributes:

- VlanTagCapable (Customer VLAN Tag capable)
- CB-Capable (frame replication and elimination capable)
- CB-StreamIdenTypeList (a list of the optional Stream Identification types supported by the interface as specified in [IEEE8021CB].)
- CB-SequenceTypeList (a list of the optional Sequence Encode/Decode types supported by the interface as specified in [IEEE8021CB].)

]]

10.3. User to Network Requirements

UserToNetworkRequirements specifies user requirements for the flow, such as latency and reliability.

The UserToNetworkRequirements object includes the following attributes:

a. NumReplicationTrees

b. MaxLatency

NumReplicationTrees specifies the number of maximally disjoint trees that the network should configure to provide packet replication and elimination for the flow. NumReplicationTrees is provided by the Source only. Destinations SHALL set this element to one. Value zero and one indicate no packet replication and elimination for the flow. When NumReplicationTrees is greater than one, packet replication and elimination is to be used for the flow. If the Source sets this element to greater than one, and packet replication and elimination is not possible in the network (e.g., no disjoint paths, or the nodes do not support packet replication and elimination), then the FailureCode of the Status object is non-zero (Section 14.1).

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 14) 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 7.2) 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?]]

11. Ingress

Placeholder ...

12. Egress

Placeholder ...

13. 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:

- The behavior of the flow (how/when it is transmited).
- The requirements of the flow from the network.
- The capabilities of the DetNet domain.

The DetNet domain object includes the following attributes:

a. DataFlowSpecification (Section 7.1)

b. TrafficSpecification (Section 7.2)

c. ServiceRank (Section 7.4)

d. DetnetDomainCapabilities (Section 13.1)

e. UserToNetworkRequirements (Section 10.3)

13.1. 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)

14. 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 7.1)

b. StatusInfo (Section 14.1)

c. AccumulatedLatency (this section below)

d. InterfaceConfiguration (Section 14.2)

e. FailedInterfaces (Section 14.3)

DataFlowSpecification identifies the flow for which status is provided. DataFlowSpecification is described in (Section 7.1) 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 10.3). If the NumReplicationTrees of the UserToNetworkRequirements (Section 10.3) is one, then the AccumlatedLatency 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 10.3) is greater than one, AccumlatedLatency SHALL provide the worst latency for all paths in use from the Source to each Destination.

14.1. 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.]]

14.2. 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 7.1)

14.3. 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.

15. Service-status

Placeholder ...

16. 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.

17. IANA Considerations

N/A.

18. Security Considerations

N/A.

19. References

19.1. 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-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>.

19.2. 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-19 (work in progress), October 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>.

[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

Balazs Varga
Ericsson
Magyar tudosok korutja 11
Budapest 1117
Hungary

Email: 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

Yuanlong Jiang
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129
China

Email: jiangyuanlong@huawei.com

Yiyong Zha
Huawei Technologies Co., Ltd.
China

Email: zhayiyong@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

X. Geng
M. Chen
Huawei
Z. Li
China Mobile
R. Rahman
Cisco Systems

October 22, 2018

DetNet Configuration YANG Model
draft-ietf-detnet-yang-00

Abstract

This document defines a YANG data model for Deterministic Networking (DetNet), which includes the DetNet topology YANG module, DetNet flow configuration YANG module and DetNet Transport QoS YANG Module.

The YANG modules in this document conform 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 April 25, 2019.

Table of Contents

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:

- 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).

- 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.

- 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.

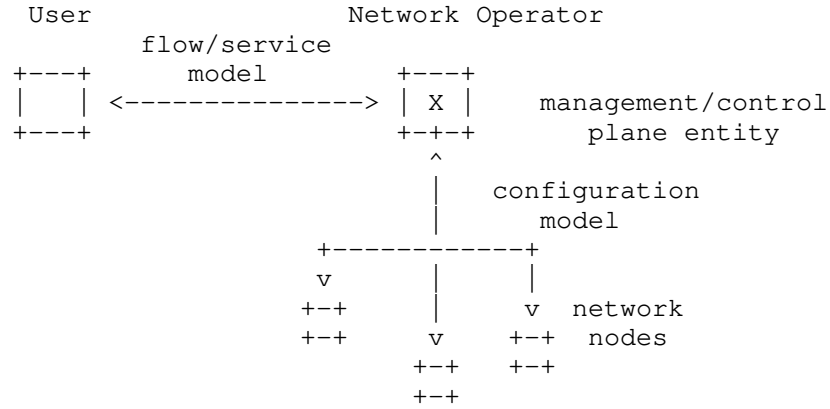


Figure 1. Three Information Models

This document defines DetNet configuration YANG [RFC7950] [RFC6991]data models, which include:

- DetNet topology model

  DetNet topology model is designed for DetNet topology/ capability discovery and device configuration, it is an augmentation of the ietf-te-toplogy model [I-D.ietf-teas-yang-te-topo]. The detail of DetNet topology model is defined in Section 3.

- DetNet flow configuration model

  DetNet flow model is designed for DetNet flow path configuration and flow status reporting. The detail of DetNet flow configuration model is defined in Section 4.

- DetNet transport QoS model

  DetNet transport QoS model is designed for QoS attributes configuration of transport tunnels to achieve end-to-end bounded latency and zero congestion loss. The detail of DetNet transport QoS model is defined in Section 5.

2. Terminologies

This documents 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 network node. The concept of DetNet node/link/LTP are similar as TE node/link/LTP that are defined in [I-D.ietf-teas-yang-te-topo].

Figure 2 shows a simple DetNet topology: A is a DetNet node, B is DetNet a LTP, and C is a DetNet link.

Figure 2. 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

attributes, which are necessary for supporting DetNet congestion protection and service protection functions:

- Bandwidth related attributes (e.g., bandwidth reserved for DetNet);
- Buffer/queue management related attributes (e.g., queue management algorithm, etc.);
- PREOF (Packet Replication, Ordering and Elimination Function) capabilities and parameters (e.g., maximum out-of-order packets, etc.);
- 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 4.1, 4.2 and 4.3, 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:

- Maximum DetNet packet processing delay
- Minimum DetNet packet processing delay
- 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:

- 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.

- 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.

- Reserved DetNet bandwidth: the bandwidth that has been reserved for DetNet flows.

- 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 :

- In-order delivery function: defined in Section 3.2.2.1 of [I-D.ietf-detnet-architecture]
- Packet replication function: defined in Section 3.2.2.2 of [I-D.ietf-detnet-architecture]
- 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:

- in-order-capability: indicates whether a LTP has the in-order delivery capability.
- 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.
- replication-capability: indicates whether a LTP has the packet replication capability.
- 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:

- queuing-algorithm-capabilities: it is modeled as a list that includes all queuing algorithms that a LTP supports.
- detnet-queues: it's modeled as a list that includes all queues of a DetNet LTP. For each queue, it has the following attributes:
- 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.

- queue-buffer-size: the size of a queue with unit of bytes.
- enabled-queuing-algorithm: indicates what queuing management algorithm is enabled.
- maximum-queuing-delay: the maximum queuing delay that a packet will undergo when transmitted through the queue.
- minimum-queuing-delay: the minimum queuing delay that a packet will undergo when transmitted through the queue.
- maximum-queuing-delay-variation: the maximum queuing delay variation that a packet will undergo when transmitted the queue.

The DetNet LTP attributes are modeled with 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 Flow 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 4.1, 4.2, 4.3, respectively. Section 4.4 gives a simple example on how to use these attributes for DetNet flow configuration.

4.1. 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:

- DetNet L3 Flow Identification, refers to Section 7.1.1 of [I-D.ietf-detnet-flow-information-model]

- 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 misbehaviored flows from going into the network, which needs:

- 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-identfication)
    |  |  |  +--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
```

4.2. 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:

- DetNetwork flow identification, refers to Section 7.1.3 of [I-D.ietf-detnet-flow-information-model].

- 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.)

- Flow Rank, refers to Section 7.3 of [I-D.ietf-detnet-flow-information-model].

- Service Rank, refers to Section 7.4 of [I-D.ietf-detnet-flow-information-model].

- Service encapsulation, refers to Section 6.2 of [I-D.ietf-detnet-dp-sol-mpls]

- 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-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-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
```

4.3. 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:

- 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.

- 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.

4.4. DetNet Flow Configuration Example

This section gives an example about how to implement an end-2-end DetNet service with the collaboration of DetNet proxy, service and transport layers.

To simplify the explanation, several terms are introduced. This document defines DetNet Service Proxy Instance (DSPI), DetNet Service Instance (DSI) and DetNet Transport Instance for end-to-end DetNet flow configuration as showed in Figure 4. DSPI 1 at Edge Node 1 (E1) maps an application flow to a DetNet Flow (DF1), which is transmitted over a DetNet tunnel (Tn1). In DSI 2 of Relay Node 1 (R1), DetNet Flow 1(DF1) was replicated into two member flows: DetNet Flow 2 (DF2) transmitted by DetNet Tunnel 2 (Tnl2) and DetNet Flow 3 (DF3) by DetNet Tunnel 3(Tnl3). In DSPI 3 of Edge Node 2 (E2), DetNet Flow 2 (DF2) and DetNet Flow 3(DF3) were merged and mapped to application flow used by CE2.

DF: DetNet Flow
DSPI: DetNet Service Proxy Instance
DSI: DetNet Service Instance
DTI: DetNet Transport Instance
Tnl: Tunnel

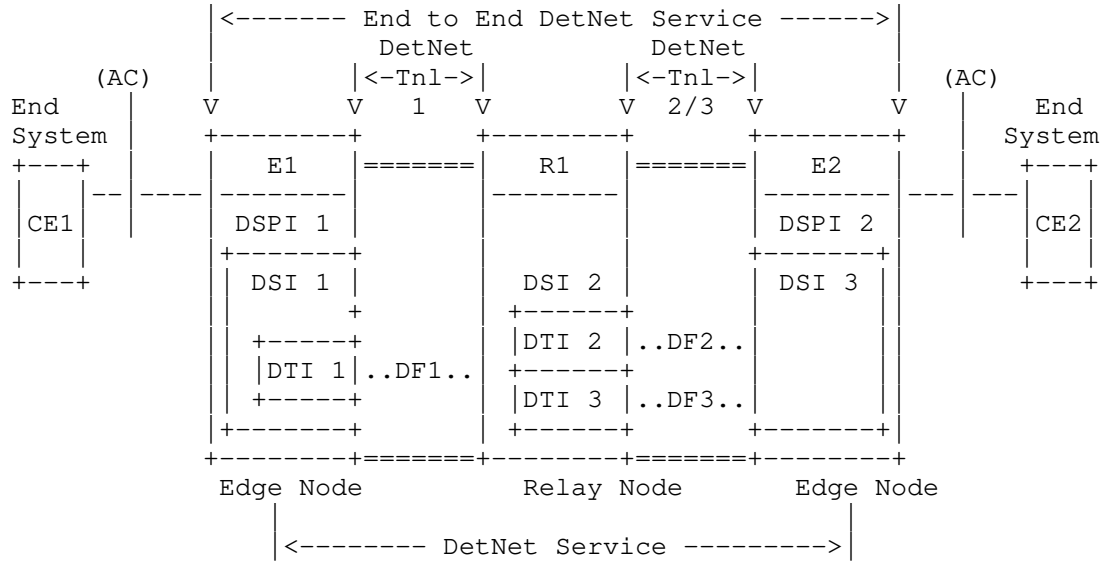


Figure 3. End-to-end DetNet Flow Configuration

5. DetNet Transport QoS Model

The YANG data model of transport QoS is very important to achieve end-to-end bounded latency and zero congestion loss. There are three possible methods to deal with the DetNet transport QoS YANG:

1. DetNet service is not aware of any QoS/queuing/bounded-latency information, and all relative parameters are defined in separate YANG models;

2. DetNet service is not aware of any of Qos/queuing/bounded-latency information, but it should maintain an interface to the corresponding YANG models;

3. DetNet service should be aware of the Qos/queuing/bounded-latency information, because some Qos/queuing/bounded-latency mechanisms are required to be configured with flow information;

How to define transport QoS YANG is still under discussion and the transport QoS YANG model is not included in the current version of the draft.

[Editor notes: more comments and inputs need here].

6. DetNet Yang Structure

6.1. DetNet Topology Model Tree Diagram

```
  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
```

6.2. DetNet Flow Configuration Model Tree Diagram

```
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-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-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-identfication)
|                  |  |  |  +--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-la
bel
          |      |  |      +--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-defi
nition
          |               +--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-identfication)
   |  |  |  +--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-defi
nition
                        +--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
```

7. DetNet YANG Model

7.1. DetNet Topology YANG Model

```
  <CODE BEGINS> file "ietf-detnet-topology@20180910.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>
```

7.2. DetNet Flow Configuration YANG Model

```
   <CODE BEGINS> file "ietf-detnet-flow@20180910.yang"
   module ietf-detnet-flow-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 pcp {
        //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-identfication {
            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>
```

8. 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.

8.1. 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.

8.2. 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.

8.3. 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 coverer 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.

9. Open issues

There are some open issues that are still under disccusion:

- 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.

- How to suppport DetNet OAM is TBD.

These issues will be resolved in the following versions of the draft.

10. IANA Considerations

This document makes no request of IANA.

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

11. Security Considerations

<TBD>

12. Acknowledgements

13. References

13.1. Normative References

[I-D.finn-detnet-bounded-latency]
Finn, N., Boudec, J., Mohammadpour, E., Varga, B., and J. Farkas, "DetNet Bounded Latency", draft-finn-detnet-bounded-latency-01 (work in progress), July 2018.

[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-flow-information-model]
Farkas, J., Varga, B., rodney.cummings@ni.com, r., Jiang, Y., and Y. Zha, "DetNet Flow Information Model", draft-ietf-detnet-flow-information-model-01 (work in progress), March 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>.

13.2. 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-02 (work in progress), March 2018.

[I-D.ietf-detnet-use-cases] Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-19 (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-16 (work in progress), July 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

Email: gengxuesong@huawei.com

Mach(Guoyi) Chen
Huawei

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: Standards Track

Y. Jiang
N. Finn
Huawei
J. Ryoo
ETRI
B. Varga
Ericsson
L. Geng
China Mobile

Expires: April 2019 October 22, 2018

Deterministic Networking Application in Ring Topologies
draft-jiang-detnet-ring-02

Abstract

Deterministic Networking (DetNet) provides a capability to carry data flows for real-time applications with extremely low data loss rates and bounded latency. This document describes how DetNet can be used in ring topologies to support Point-to-Point (P2P) and Point-to-Multipoint (P2MP) real-time services.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on April 22, 2019.

Table of Contents

1. Introduction

An overview of Deterministic Networking (DetNet) architecture is given in [I-D.ietf-detnet-architecture], and DetNet data plane encapsulations are specified in [I-D.ietf-detnet-dp-sol]. But there is not any discussion on a ring topology in [I-D.ietf-detnet-architecture] yet. Furthermore, [I-D.ietf-detnet-use-cases] outlines several Detnet use cases where multicast capability is needed. If a multicast service replicates all of its packets from the source (as a traditional Virtual Private LAN Service (VPLS) does), the requirements of deterministic delay and high availability for all these replicated packets will pose a great challenge to the Detnet network.

In fact, ring topologies have been very popular and widely deployed in network arrangements for various transport networks, such as Synchronous Digital Hierarchy, Synchronous Optical Network, Optical Transport Network, and Ethernet. The IETF has done some work on ring protection in Multi-Protocol Label Switching - Transport Profile (MPLS-TP), such as [RFC6974] and [RFC8227]. All these works, except Ethernet ring protection, typically use swapping or steering as the protection mechanism. As ring topologies are widely deployed for transport networks, it is also necessary for DetNet to support ring topologies (currently, there is not any discussion on a ring topology in [I-D.ietf-detnet-architecture] yet).

This draft demonstrates how DetNet can be used in a ring topology. Specifically, DetNet ring supports for Point-to-Point (P2P) and Point-to-Multipoint (P2MP, for multicast services) are discussed in details. This document assumes that MPLS encapsulation for DetNet is supported as specified in [I-D.ietf-detnet-dp-sol-mpls] and all nodes in a ring network can support the Multi-Protocol Label Switching (MPLS) functionalities. It should be noted that it is more convenient for DetNet to support a ring topology with the intrinsic duplication and elimination mechanism, as there is no need of swapping or steering operations (consequently, its Operations, Administration and Maintenance (OAM) can also be simplified) for any service protection.

1.1. 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].

1.2. Terminology

DetNet Deterministic Networking

LSP Label Switched Path

MPLS Multi-Protocol Label Switching

MPLS-TP Multi-Protocol Label Switching - Transport Profile

P2MP Point-to-Point

P2P Point-to-Multipoint

PEF Packet Elimination Function, see [I-D.ietf-detnet-architecture]

POF Packet Ordering Function, see [I-D.ietf-detnet-architecture]

PRF Packet Replication Function, see [I-D.ietf-detnet-architecture]

PW Pseudowire

2. P2P DetNet Ring

2.1. DetNet applications on a single ring for P2P traffic

Figure 1 depicts an example of the DetNet ring for P2P real time traffic. Nodes A and C are DetNet aware devices, and P2P DetNet traffic is transported from node A to node C.

A clockwise and a counter clockwise Pseudowire (PW, or S-Label as described in [I-D.ietf-detnet-dp-sol-mpls]) and Label Switched Path (LSP, or T-Label as described in [I-D.ietf-detnet-dp-sol-mpls]) tunnel are configured from node A to node C respectively. The DetNet traffic is replicated by a Packet Replication Function (PRF) in node A, encapsulated with the specific PW and LSP labels, and transported on both LSP paths towards node C. Upon reception of the traffic, node C terminates the LSP and is aware of the DetNet traffic by inspection of the PW label carried in each packet. A Packet Elimination Function (PEF) in node C guarantees that only one copy of the DetNet service exits on egress with the help of the DetNet sequence number. A Packet Ordering Function (POF) can further re-order packets in node C before transport of these packets to the destination.

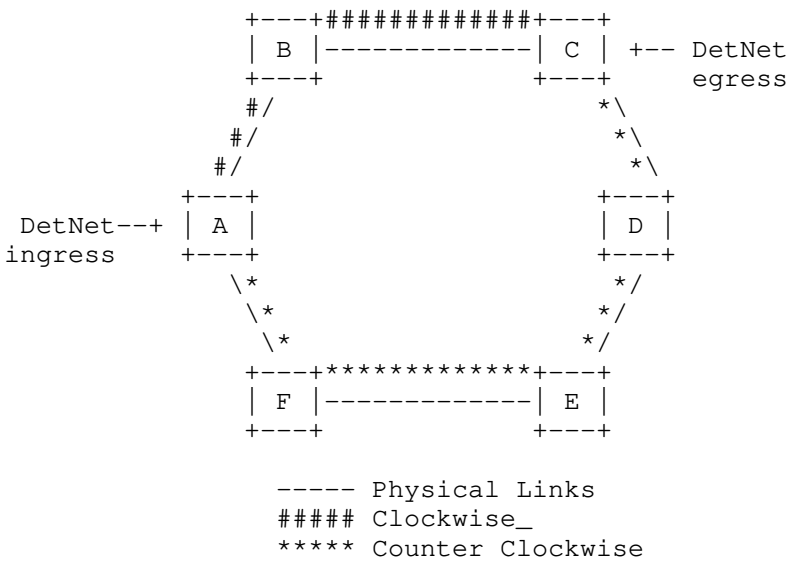


Figure 1: DetNet Ring for P2P traffic

2.2. Implementation implications of a DetNet ring for P2P traffic

In a DetNet ring for P2P traffic, one path may be far longer than the other path for the DetNet (this is a DetNet issue more general than a ring).

The buffer needs to be large enough to accommodate for the sequence number difference between these two paths. Otherwise, some packets may get lost when a link fault causes traffic switching from a path to another path.

3. P2MP DetNet Ring

3.1. DetNet applications on a single ring for P2MP traffic

Figure 2 further depicts an example of the DetNet ring for P2MP real time traffic. Nodes A, B, C, E and F are DetNet aware devices, and P2MP DetNet traffic is transported from head-end node A to multiple tail-end nodes C, E and F.

Two approaches are described in Section 3.2 and 3.3 for P2MP traffic.

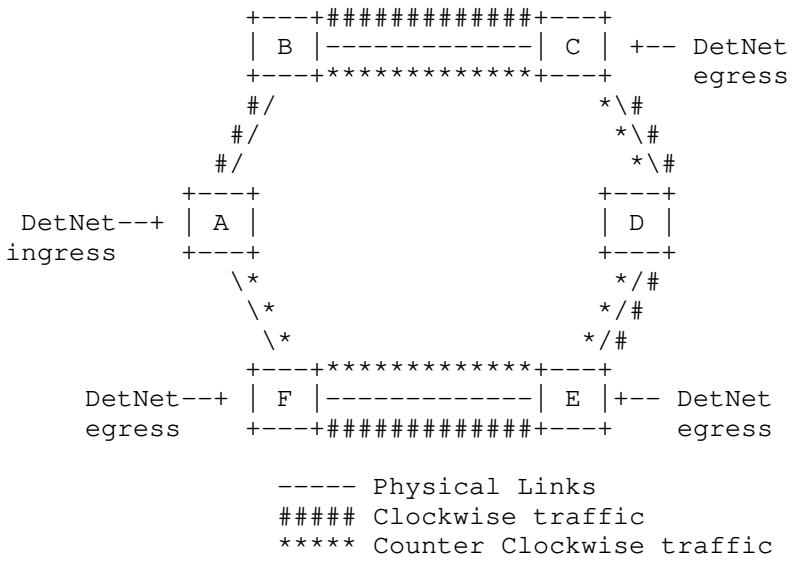


Figure 2: DetNet Ring for P2MP traffic

3.2. Section LSPs as underlay (Service layer replication)

If section LSPs are used as an underlay for DetNet services, a bidirectional section LSP tunnel is set up between each pair of neighboring nodes in the ring (e.g., node A and node B, ..., node F and node A). In this case, DetNet PW layer replicates the DetNet packets from one tail-end to another neighboring tail-end.

The DetNet head-end (i.e., node A) in the ring needs to support DetNet replication function. Upon reception on node A, the DetNet traffic is replicated in node A, encapsulated with the specific PW and section LSP labels, and then transported on both section LSPs (i.e., A-B and A-F) originated from the head-end.

All intermediate nodes (non tail-ends) on the ring SHOULD transparently forward the DetNet traffic with a specific PW to the next hop on the ring in the same direction.

All DetNet tail-ends except the penultimate node (egress nodes such as nodes C and E in the clockwise, and node F, E and C in the counter clockwise) on the ring MUST support both DetNet PRF and PEF functions, and MAY further support a DetNet POF function. For the example of Figure 2, upon reception of the clockwise traffic, node C terminates the section LSP and is aware of the DetNet traffic by

inspection of the PW label in the packet. Firstly, node C needs to transparently forward the DetNet traffic with a specific PW to the next hop on the ring in the same direction. Secondly, DetNet traffic is directed to a DetNet PEF associated with a specific PW, only one copy of the DetNet service is selected by inspection of the DetNet sequence number. Furthermore, if DetNet POF function is enabled, the packets in the DetNet flow are reordered before exit to DetNet egress.

If multiple endpoints are attached to a tail-end node, a multicast module can be used to forward the filtered DetNet traffic to all these endpoints.

To avoid a loop of DetNet service, the penultimate node in the ring (such as node B on the counter clock-wise LSP) needs to terminate the DetNet flow. For example, upon reception of the clockwise DetNet traffic, node F terminates the DetNet traffic by inspection of the PW label in the packet. As an alternative, the last DetNet tail-end (such as node C on the counter clock-wise LSP) may terminate the DetNet flow, so that the bandwidth from this node to the penultimate node can be saved.

3.3. P2MP LSP tunnels as underlay (LSP layer replication)

If P2MP LSPs are used as an underlay for the DetNet service, a P2MP unidirectional LSP tunnel in clockwise is set up from head-end (ingress node A) to all the tail-ends (egress nodes C, E and F) for the ring, and another P2MP unidirectional LSP tunnel in counter clockwise is set up from head-end (ingress node A) to all the tail-ends (egress nodes F, E and C) for the ring. Thus, a PRF in LSP layer replicates the DetNet packets from one tail-end to another neighboring tail-end.

The DetNet head-end (i.e., node A) in the ring needs to support DetNet PRF function. Upon reception on node A, the DetNet traffic is replicated, encapsulated with the specific PW and P2MP LSP labels, and transported on both P2MP LSP tunnels in the ring.

All DetNet tail-ends (egress nodes such as node C, E and F in Figure 2) on the ring need to support the DetNet PEF function. For example, upon reception of the traffic, node C pops the P2MP LSP label and is aware of the DetNet traffic by inspection of the PW label in the label stack. Traffic from both directions with the same PW is directed to the same PEF so that only one copy of the DetNet service is selected by inspection of the DetNet sequence number. Furthermore, if DetNet POF function is enabled, the packets in the DetNet flow are reordered before exit to DetNet egress.

If multiple endpoints are attached to a tail-end node, a multicast module can be used to forward the filtered DetNet traffic to all these endpoints.

4. DetNet Ring Interconnections

Two DetNet rings can be connected via one or more interconnection nodes. Figures 3(a) and 3(b) show the ring interconnection scenarios with a single node and dual nodes respectively. In the interconnected rings, each ring operates in the same way as described in Sections 2 and 3 except the node or nodes that are used to interconnect two rings.

In this section, we describe the behavior of interconnection nodes with the traffic going from Ring L to Ring R. Symmetrical description is assumed for the traffic in the other direction (i.e., from Ring R to Ring L).

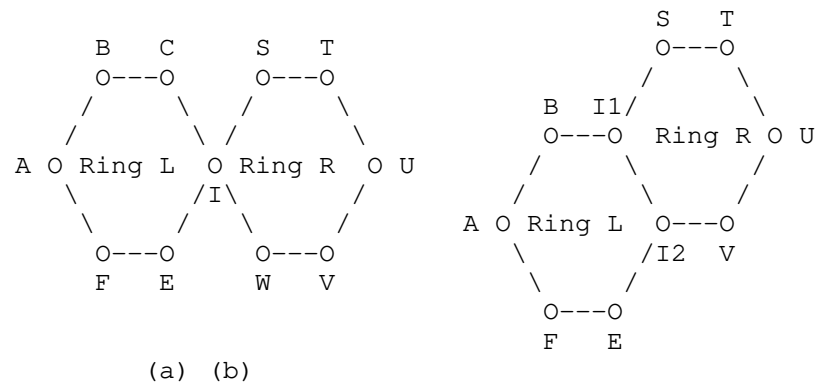


Figure 3: DetNet ring interconnection with: (a) single node (node I), and (b) dual nodes (nodes I1 and I2).

4.1. Single node interconnection

In the case of the single node interconnection, as shown in Figure 3(a), both P2P and P2MP DetNet traffic that needs to be transported between Ring L and Ring R uses a single interconnection node between two rings. Thus, the interconnection node acts as a DetNet relay node, which provides both PRF and PEF functions.

For P2P DetNet traffic going from Ring L to Ring R, interconnection node I receives the same Detnet flow traffic from both node C and node E (i.e., clockwise and counter-clockwise), a PEF in node I performs packet elimination, and a PRF in node I replicates the packet, node I then sends one copy to node S and another copy to node W.

For P2MP DetNet traffic going from Ring L to Ring R, interconnection node I performs the same packet elimination and replication functions as described above. In addition, node I further transparently forwards the P2MP DetNet traffic on Ring L in the same direction if it is not the last tail-end node.

4.2. Dual node interconnection

In order to prevent a single point of failure, two interconnection nodes can be used as shown in Figure 3(b). To provide high availability for DetNet services, dual node interconnection is recommended. Two interconnection nodes act as DetNet relay nodes, each provides both packet replication and elimination functions.

4.2.1. Dual node interconnection for P2P traffic

For the P2P DetNet traffic that flows from Ring L to Ring R, the operation of interconnection nodes I1 and I2 follows the description on relay nodes shown in Figure 1 of Section 3.2.4 in [I-D.ietf-detnet-architecture]. In the following, the operation is explained with Figure 3(a).

When interconnection node I1 receives clockwise traffic from node B, it replicates the traffic and sends one copy to interconnection node I2 and another copy to a PEF in node I1.

When interconnection node I1 receives counter-clockwise traffic from interconnection node I2, it also forwards the traffic to the PEF of I1.

At the PEF of I1, duplicate elimination is performed for the clockwise traffic from node B and the counter-clockwise traffic from interconnection node I2, and only one copy is sent to the clockwise direction of Ring R (i.e., sent towards node S). Furthermore, if DetNet POF function is enabled on I1, the packets in the DetNet flow are reordered before exit to Ring R.

When interconnection node I2 receives counter-clockwise traffic from node E, it replicates the traffic and sends one copy to interconnection node I1 and another copy to a PEF in node I2.

When interconnection node I2 receives clockwise traffic from interconnection node I1, it also forwards the traffic to the PEF of I2.

At the PEF of I2, duplicate elimination is performed for the counter-clockwise traffic from node E and the clockwise traffic from interconnection node I1, and only one copy is sent to the counter-clockwise direction of Ring R (i.e., sent towards node V). Furthermore, if DetNet POF function is enabled on node I2, the packets in the DetNet flow are reordered before exit to Ring R.

4.2.2.Dual node interconnection for P2MP traffic using section LSP

For the P2MP traffic that flows from Ring L to Ring R, each ring is configured and operated as described in Section 3.2 except the interconnection nodes, whose operations are described below.

When interconnection node I1 receives clockwise traffic from node B, its PRF replicates the traffic and sends one copy to interconnection node I2 and another copy to node I1’s PEF.

When interconnection node I1 receives the counter-clockwise traffic from interconnection node I2, its PRF replicates the traffic and sends one copy to node B and another copy to node I1’s PEF unless interconnection node I1 is the penultimate node for the counter-clockwise traffic on Ring L. In the case that interconnection node I1 is the penultimate node for the counter-clockwise traffic on Ring L, the counter-clockwise traffic from interconnection node I2 is only forwarded to node I1’s PEF.

At node I1’s PEF, duplicate elimination is performed for the clockwise traffic from node B and the counter-clockwise traffic from interconnection node I2, and only one copy is sent to the clockwise direction of Ring R (i.e., sent towards node S). Furthermore, if DetNet POF function is enabled on node I1, the packets in the DetNet flow are reordered before exit to Ring R.

When interconnection node I2 receives the counter-clockwise traffic from node E, its PRF replicates the traffic and sends one copy to interconnection node I1 and another copy to node I2’s PEF.

When interconnection node I2 receives the clockwise traffic from interconnection node I1, its PRF replicates the traffic and sends one copy to node E and another copy to node I2’s PEF unless interconnection node I2 is the penultimate node for the clockwise

traffic in Ring L. In the case that interconnection node I2 is the penultimate node for the clockwise traffic in Ring L, the clockwise traffic from interconnection node I1 is only forwarded to node I2's PEF.

At node I2's PEF, duplicate elimination is performed for the counter-clockwise traffic from node E and the clockwise traffic from interconnection node I1, and only one copy is sent to the counter-clockwise direction of Ring R (i.e., sent towards node V). Furthermore, if DetNet POF function is enabled on node I2, the packets in the DetNet flow are reordered before exit to Ring R.

4.2.3.Dual node interconnection for P2MP traffic using P2MP LSP

If P2MP LSPs are used in the interconnected rings, two P2MP unidirectional LSP tunnels are used on each ring for the clockwise and counter-clockwise directions.

When the P2MP traffic is forwarded from one ring to another ring, for example from Ring L to Ring R in Figure 3(b), each P2MP LSP in Ring L MUST include interconnection nodes I1 and I2 as its tail-ends. For Ring R, one P2MP LSP is set up from interconnection node I1 to all the tail-ends in the clockwise direction on Ring R, and the other P2MP LSP is set up from interconnection node I2 to all the tail-ends in the counter-clockwise direction on Ring R. Therefore, an interconnection node acts as a tail-end for one ring and a head-end for another ring in one direction, and performs the same operation of tail-end and head-end as specified in Section 3.3.

5. Resource reservation

In order to guarantee that DetNet flows don't suffer from network congestion, resource reservation considerations as outlined in Section 4.3.2 of [I-D.ietf-detnet-architecture] apply here.

6. Security Considerations

This document describes the application of DetNet on general ring topologies. Thus the security considerations as described in [I-D.ietf-detnet-dp-sol] also apply to this document.

7. IANA Considerations

There are no IANA actions required by this document.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997

[I-D.ietf-detnet-architecture] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture (work in progress), June 2018

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

8.2. Informative References

[I-D.ietf-detnet-dp-sol] Korhonen, J., Andersson, L., Jiang, Y., and etc., "DetNet Data Plane Encapsulation", draft-ietf-detnet-dp-sol (work in progress), March 2018

[I-D.ietf-detnet-use-cases] Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases (work in progress), October 2018

[RFC6974] Weingarten, Y., Bryant, S., and etc., "Applicability of MPLS Transport Profile for Ring Topologies", RFC 6974, July 2013

[RFC8227] Cheng, W., Wang, L., and etc., "MPLS-TP Shared-Ring Protection (MSRP) Mechanism for Ring Topology", RFC 8227, August 2017

9. Acknowledgments

The authors would like to thank Loa Anderson for his discussion.

Authors' Addresses

Yuanlong Jiang
Huawei Technologies Co., Ltd.
Bantian, Longgang district
Shenzhen 518129, China
Phone: +86-18926415311
Email: jiangyuanlong@huawei.com

Norman Finn
Huawei Technologies Co. Ltd
3755 Avocado Blvd,
California 91941, USA
Phone: +1 925 980 6430
Email: norman.finn@mail01.huawei.com

Jeong-dong Ryoo
ETRI
218 Gajeongno
Yuseong-gu, Daejeon 305-700, South Korea
Phone: +82-42-860-5384
Email: ryoo@etri.re.kr

Balazs Varga
Ericsson
Konyves Kalman krt. 11/B
Budapest 1097
Hungary
Email: balazs.a.varga@ericsson.com

Liang Geng
China Mobile
Beijing, China
Email: gengliang@chinamobile.com

DetNet Working Group
Internet-Draft
Intended status: Informational
Expires: December 4, 2018

G. Mirsky
ZTE Corp.

June 2, 2018

Operations, Administration and Maintenance (OAM) for Deterministic Networks (DetNet)
draft-mirsky-detnet-oam-00

Abstract

This document lists functional requirements for Operations, Administration and Maintenance (OAM) toolset in Deterministic Networks (DetNet) and, using these requirements, and analyzes possible DetNet data plane solutions.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 4, 2018.

Copyright Notice

Table of Contents

1. Introduction

[I-D.ietf-detnet-architecture] introduces and explains Deterministic Networks (DetNet) architecture and how the Packet Replication and Elimination function (PREF) can be used to ensure low packet drop ratio in DetNet domain.

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

This document lists the functional requirements toward OAM for DetNet domain. The list can further be used to for gap analysis of available OAM tools to identify possible enhancements of existing or whether new OAM tools are required to support proactive and on-demand path monitoring and service validation.

2. Conventions used in this document

2.1. Terminology

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

AC Associated Channel

CW Control Word

DetNet Deterministic Networks

d-CW DetNet Control Word

OAM: Operations, Administration and Maintenance

PREF Packet Replication and Elimination Function

PW Pseudowire

RDI Remote Defect Indication

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

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

2.2. Keywords

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

3. Requirements

This section lists requirements for OAM in DetNet domain:

1. The listed requirements MUST be supported with any type of underlay network over which a DetNet domain can be realized.

2. It MUST be possible to initiate DetNet OAM session from any DetNet node towards another DetNet node(s) within given domain.

3. It SHOULD be possible to initialize DetNet OAM session from a centralized controller.

4. DetNet OAM MUST support proactive and on-demand OAM monitoring and measurement methods.

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

6. DetNet OAM MUST support unidirectional OAM methods, continuity check, connectivity verification, and performance measurement.

7. DetNet OAM MUST support bi-directional OAM methods. Such OAM methods MAY combine in-band monitoring or measurement in the forward direction and out-of-bound notification in the reverse direction, i.e. from egress to ingress end point of the OAM test session.

8. DetNet OAM MUST support proactive monitoring of a DetNet node availability in the given DetNet domain.

9. DetNet OAM MUST support Path Maximum Transmission Unit discovery.

10. DetNet OAM MUST support Remote Defect Indication (RDI) notification to the DetNet node performing continuity checking.

11. DetNet OAM MUST support performance measurement methods.

12. DetNet OAM MUST support unidirectional performance measurement methods. Calculated performance metrics MUST include but are not limited to throughput, loss, delay and delay variation metrics. [RFC6374] provides great details on performance measurement and performance metrics.

13. DetNet OAM MUST support defect notification mechanism, like Alarm Indication Signal. Any DetNet node in the given DetNet domain MAY originate a defect notification addressed to any subset of nodes within the domain.

14. DetNet OAM MUST support methods to enable survivability of the DetNet domain. These recovery methods MAY use protection switching and restoration.

4. DetNet Data Plane in Support of Active OAM

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

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

The data plane encapsulation for DetNet specified in [I-D.ietf-detnet-dp-sol] has been analyzed in details in [I-D.bryant-detnet-mpls-dp] and [I-D.malis-detnet-ip-dp] for use in MPLS and IP networks respectively. For the MPLS underlay network DetNet flows to be encapsulated analogous to pseudowires (PW) over MPLS packet switched network, as described in [RFC3985], [RFC4385]. Generic PW MPLS Control Word (CW), defined in [RFC4385], for DetNet displayed in Figure 1.

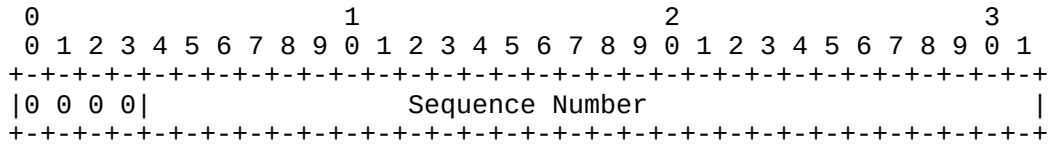


Figure 1: DetNet Control Word Format

PREF in the DetNet domain composed by a combination of nodes that perform replication and elimination sub-functions. The elimination sub-function always uses packet sequencing information, e.g., value in the Sequence Number field of DetNet CW (d-CW). The replication sub-function uses one of two options:

- use S-Label and d-CW information;
- use S-Label information.

For data packets Figure 2 presents an example of PREF in DetNet domain regardless of how the replication sub-function realized in the domain.

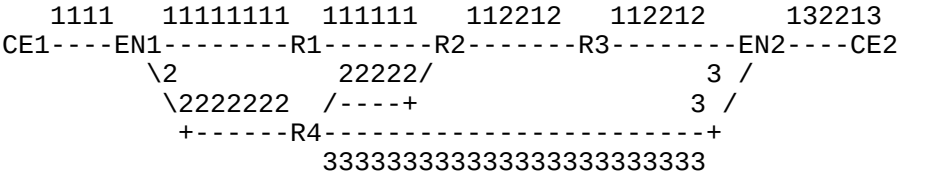


Figure 2: DetNet Data Plane Based on PW

4.1. DetNet Active OAM Encapsulation

DetNet OAM, like PW OAM, uses PW Associated Channel Header defined in [RFC4385]. Figure 3 displays encapsulation of a DetNet active OAM packet. Figure 4 displays format of the DetNet Associated Channel (AC).

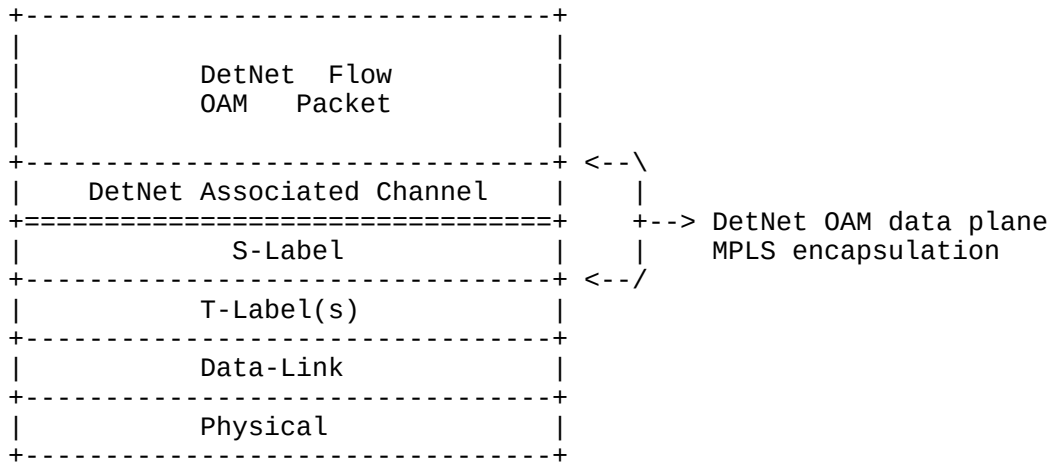


Figure 3: DetNet PW OAM Packet Encapsulation

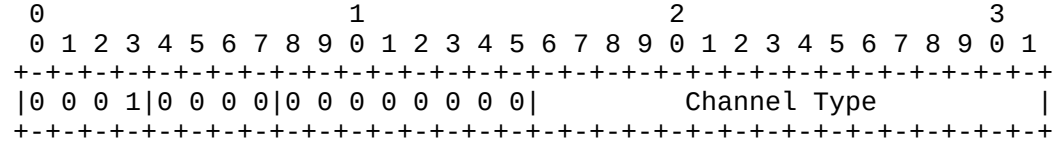


Figure 4: DetNet Associated Channel Header Format

4.2. DetNet PREF Interaction with Active OAM

Consider the scenario when EN1 injects DetNet active OAM packet with the same S-Label as the DetNet service reflected in Figure 2. EN1 is the first node with the replication sub-function. If the replication uses S-Label information and the sequencing information in d-CW (the first option), then EN1 will only forward the OAM packet without replicating it because OAM encapsulation doesn't include d-CW. The path that active OAM packet traverses through the DetNet domain presented in Figure 5 with 'O'. The figure clearly demonstrates that

the DetNet OAM packet does not traverse all the segments that are traversed by the DetNet data packet as displayed in Figure 2.

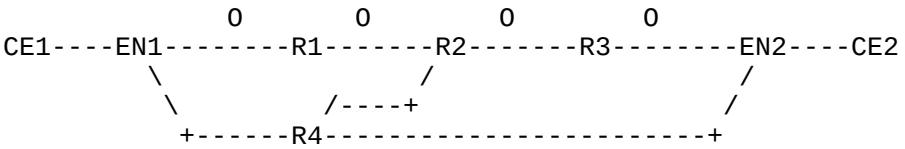


Figure 5: OAM in DetNet Data Plane Based on PW

If the replication is based solely on S-Label (the second option), EN1 node will replicate the OAM packet accordingly. The replicated packet will be processed by the replication function at R4. As result, the same OAM packet will be forwarded and another copy injected into the network. This case displayed in Figure 6. The OAM packet does traverse all links and nodes that the DetNet data packet of the monitored flow traverses but the egress node EN2 receives multiple, three in this example, copies of the same packet because the elimination function cannot be applied to the DetNet active OAM packet.

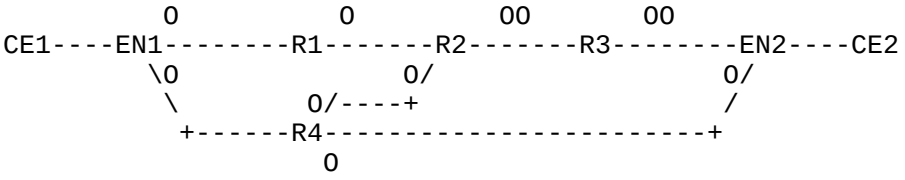


Figure 6: Over-Replication of Active OAM Packets

4.3. Alternative Encapsulation for DetNet

Introduction of DetNet header, that includes all necessary characteristic information to efficiently, among other scenarios, use multipath underlay, perform PERF, as part of DetNet service layer encapsulation allows DetNet active OAM packets to be in-band with the monitored DetNet data flow. Figure 7 presents the format of DetNet packet with MPLS encapsulation.

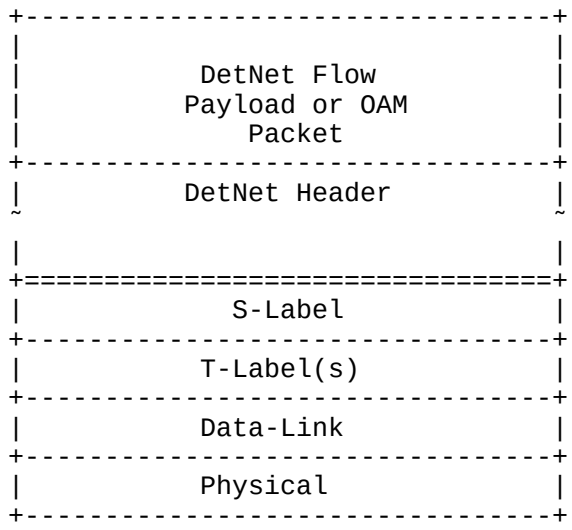


Figure 7: DetNet Packet with DetNet Header Encapsulation over MPLS Underlay

Demultiplexing of type of the payload encapsulated in the DetNet packet achieved using a field that explicitly identifies, e.g., OAM, Ethernet, or IPvX.

5. Use of Hybrid OAM in DetNet

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

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

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

6. IANA Considerations

This document does not propose any IANA consideration. This section may be removed.

7. Security Considerations

This document lists the OAM requirements for a DetNet domain and does not raise any security concerns or issues in addition to ones common to networking.

8. Acknowledgment

TBD

9. References

9.1. Normative References

[I-D.bryant-detnet-mpls-dp]
Bryant, S. and M. Chen, "Operation of Deterministic Networks over MPLS", draft-bryant-detnet-mpls-dp-00 (work in progress), March 2018.

[I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-detnet-architecture-05 (work in progress), May 2018.

[I-D.ietf-detnet-dp-sol]
Korhonen, J., Andersson, L., Jiang, Y., Finn, N., Varga, B., Farkas, J., Bernardos, C., Mizrahi, T., and L. Berger, "DetNet Data Plane Encapsulation", draft-ietf-detnet-dp-sol-04 (work in progress), March 2018.

[I-D.malis-detnet-ip-dp]
Malis, A., Bryant, S., Chen, M., and B. Varga, "DetNet IP Encapsulation", draft-malis-detnet-ip-dp-00 (work in progress), March 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>.

[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>.

9.2. Informational References

[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>.

[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>.

[RFC6374] Frost, D. and S. Bryant, "Packet Loss and Delay Measurement for MPLS Networks", RFC 6374, DOI 10.17487/RFC6374, September 2011, <https://www.rfc-editor.org/info/rfc6374>.

[RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <https://www.rfc-editor.org/info/rfc7799>.

[RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <https://www.rfc-editor.org/info/rfc8321>.

Author’s Address

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 31, 2019

L. Qiang, Ed.
B. Liu
T. Eckert, Ed.
Huawei
L. Geng
L. Wang
China Mobile

September 27, 2018

Large-Scale Deterministic Network
draft-qiang-detnet-large-scale-detnet-02

Abstract

This document presents the framework and key methods for Large-scale Deterministic Networks (LDN). It achieves scalability for the number of supportable deterministic traffic flows via Scalable Deterministic Forwarding (SDF) that does not require per-flow state in transit nodes and precise time synchronization among nodes. It achieves Scalable Resource Reservation (SRR) by allowing for it to be decoupled from the forwarding plane nodes, and aggregating resource reservation status in time slots.

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 March 31, 2019.

Table of Contents

1. Introduction

Deploying deterministic service over large-scale network will face some technical challenges, such as

- massive number of deterministic flows vs. per-flow operation and management;
- long link propagation may bring in significant jitter;
- time synchronization is hard to be achieved among numerous devices, etc.

Motivated by these challenges, this document presents a Large-scale Deterministic Network (LDN) system, which consists of Scalable Deterministic Forwarding (SDF) at forwarding plane and Scalable Resource Reservation (SRR) at control plane. The technologies of SDF and SRR can be used independently.

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. This document 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", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

1.2. Terminology & Abbreviations

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

TSN: Time Sensitive Network

CQF: Cyclic Queuing and Forwarding

LDN: Large-scale Deterministic Network

SDF: Scalable Deterministic Forwarding

SRR: Scalable Resource Reservation

DSCP: Differentiated Services Code Point

EXP: Experimental

TC: Traffic Class

T: the length of a cycle

H: the number of hops

K: the size of aggregated resource reservation window

2. Overview

2.1. Summary

The Large-Scale Deterministic Network solution (LDN) consists of two parts: The Scalable Deterministic Forwarding Plane (SDF) as its forwarding plane and the Scalable Resource Reservation (SRR) as its control plane. In the SDF, nodes in the network have synchronized frequency, and each node forwards packets in a slotted fashion based on a cycle identifiers carried in packets. Ingres nodes or senders have a function called gate to shape/condition traffic flows. Except for this gate function, the SDF has no awareness of individual flows. The SRR maintains resource reservation states for deterministic flows, Ingress nodes maintain per-flow states and core nodes aggregate per-flow states in time slots.

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 (without picture), where N 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 links bandwidth, one from each receiving interface to the outgoing interface.

Node N 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 N is uncontrolled, then the worst case is for them to all arrive at the same time. One packet has to wait in N 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 N2 downstream from N, this problem can become worse. Assume N2 has 10 upstream nodes like N,

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 physics of the forwarding plane across the node, but not the accumulated characteristics of prior hop traffic profiles.

2.2.3. Cyclic Forwarding

The common approach to solve that problem is that of a cyclic hop-by-hop forwarding mechanism. Assume packets forwarded from N1 via N2 to N3 as shown in Figure 1. When N1 sends a packet P to interface I1 with a Cycle X, it must be guaranteed by the forwarding mechanism that N2 will forward P via I2 to N3 in a cycle Y.

The cycle of a packet can either be deduced by a receiving node from the exact time it was received as is done in SDN/TDMA systems, and/or it can be indicated in the packet. This document solution relies on such markings because they allow to reduce the need for synchronous hop-by-hop transmission timings of packets.

In a packet marking based slotted forwarding model, node N1 needs to send packets for cycle X before the latest possible time that will allow for N2 to further forward it in cycle Y to N3. Because of the marking, N1 could even transmit packets for cycle X before all packets for the previous cycle (X-1) have been sent, reducing the synchronization requirements between across nodes.

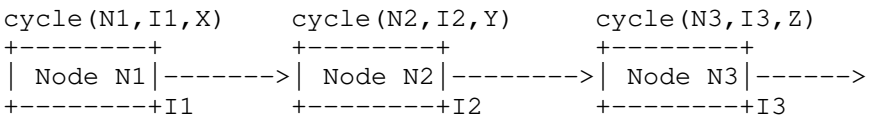


Figure 1: Cyclic Forwarding

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 SDF is deployed together with such non-deterministic delay traffic than 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 2 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 Scalable Deterministic Forwarding (SDF). This forwarding requires no per-flow state on the nodes.

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.

A Scalable Resource Reservation (SRR) works as control plane. It records reserved resources for deterministic flows. Per-flow state is maintained on the ingress edge node, and aggregated state is maintained on core node.

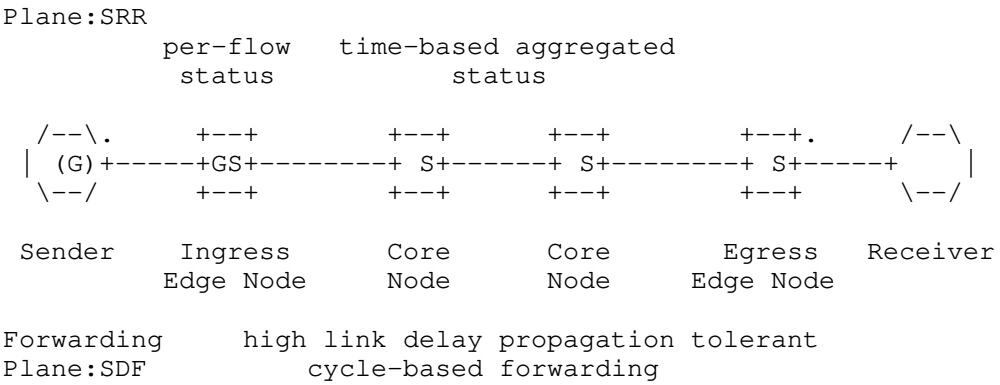


Figure 2: System Overview

3. Scalable Deterministic Forwarding

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 3.

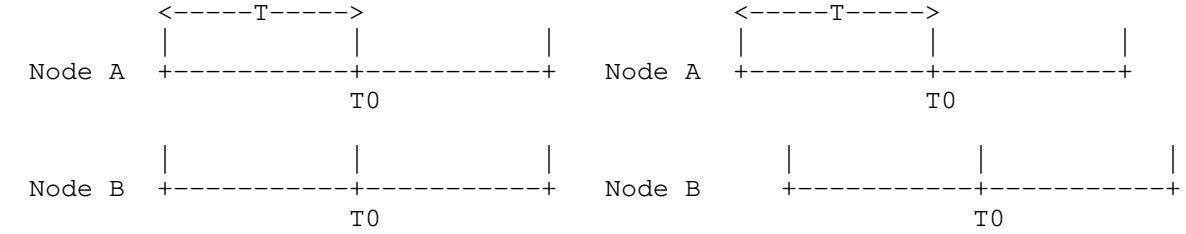


Figure 3: Time Synchronization & Clock Synchronization

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 Scalable Deterministic Forwarding (SDF) permits frequency synchronization and link propagation delay may exceed T. Besides these two points, CQF and the asynchronous forwarding of SDF are very similar.

Figure 4 compares CQF and SDF 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. Due to long link propagation delay and frequency synchronization, Node B will receive packets from Node A at different cycle denoted by y in the SDF, and Node B swaps the cycles carried in packets with y+1, then sends out those packets at cycle y+1. This cycle mapping (e.g., x --> y+1) can be realized as an adjustment value, and it exists between any pair of neighbor nodes. With this 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.

In right part of Figure 4, Node A sends a packet with cycle identifier x in cycle x indicated by the identifier. After received by Node B, the cycle identifier x in the packet will be modified by the adjustment value to get a new cycle identifier y+1. Then the identifier y+1 will replace the original identifier x. Finally, the packet with the cycle identifier y+1 will be sent by Node B in cycle y+1 indicated by the new identifier.

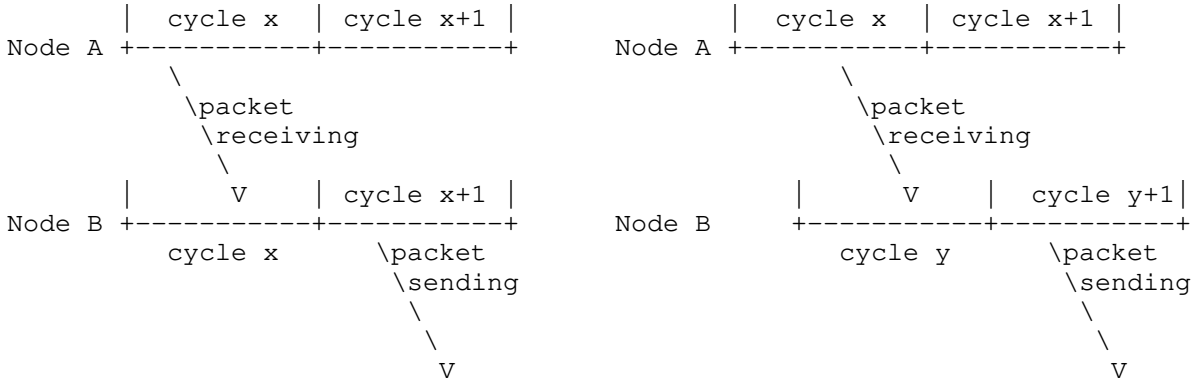


Figure 4: CQF & SDF

3.1. Three Queues

In CQF each port needs to maintain 2 (or 3) queues: one is used to buffer newly received packets, another one 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 SDF, at least 3 cyclic queues are maintained for each port on a node. A cyclic queue corresponds to a cycle.

As Figure 5 illustrated, a node may receive packets sent at two different cycles from a single upstream node due to the absence of time synchronization. Following the cycle mapping (i.e., x --> 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, two queues are needed to store the newly received packets, as well as one queue to store the sending packets. In order to absorb more link delay variation (such as on radio interface), more queues may be necessary.

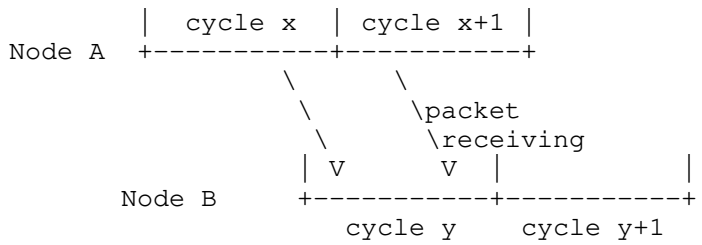


Figure 5: Three Queues in SDF

3.2. Cycle Mapping

When this packet is received by Node B, some methods are possible how the forwarding plane could operate. In one method, Node B has a mapping determined by the control plane. Packets from (the link from) Node A indicating cycle x are mapping into cycle y+1. This mapping is necessary, because all the packets from one cycle of the sending node need to get into one cycle of the receiving node. This is called "configured cycle mapping".

Instead of configuring an explicit cycle mapping such as cycle x -> cycle y+1, the receiving Node B could also have the intelligence in the forwarding plane to recognize the first packet from (the link from) Node A that has a new cycle x number, and map this cycle x to a cycle y after the current cycle. We call this option "self synchronized cycle mapping".

3.2.1. Cycle Identifier Carrying

In self synchronized cycle mapping, cycle identifier needs to be carried in the SDF packets, so that an appropriate queue can be selected accordingly. That means 2 bits are needed in the three queues model of SDF, in order to identify different cycles between a pair of neighboring nodes. There are several ways to carry this 2 bits cycle identifier. This document does not yet aim to propose one, but gives an (incomplete) list of ideas:

- DSCP of IPv4 Header
- Traffic Class of IPv6 Header
- TC of MPLS Header (used to be EXP)
- EtherType of Ethernet Header
- IPv6 Extension Header

- o TLV of SRv6
- o TC of MPLS-SR Header (used to be EXP)
- o Three labels/adjacency SIDs for MPLS-SR

4. Scalable Resource Reservation

SDF must work with some resource reservation mechanisms, that can fulfill the role of the Scalable Resource Reservation (SRR). This resource reservation guarantees the necessary network resources (e.g., bandwidth) when deterministic flows are scheduled including the slots through which the traffic travels hop-by-hop. Network nodes have to record how many network resources are reserved for a specific flow from when it starts to when it ends (e.g., <flow_identifier, reserved_resource, start_time, end_time>). Maintaining per-flow resource reservation state may be acceptable to edge nodes, but un-acceptable to core nodes. [draft-ietf-detnet-architecture] pointed out that aggregation must be supported for scalability.

SRR aggregates per-flow resource reservation states in each time slot following the steps:

1. Dividing time into time slots. Then the per-flow resource reservation message can be expressed as <flow_identifier, reserved_resource, start_time_slot, num_time_slot> accordingly, where flow_identifier is the identifier of a deterministic flow, reserved_resource indicates how much resource is reserved, start_time_slot is the number of time slot from which resource reservation starts (e.g., the time slot that a new resource reservation request generates), num_time_slot indicates how many time slots the resource will be reserved. Note that time slot here is irrelevant to the cycle in SDF.

2. Edge node still maintains per-flow resource reservation states. While core node calculates and maintains the sum of reserved_resources (or remaining resources) of each time slot. That is a core node just needs to maintain a variable for each time slot. A core node can maintain K time slots' resource reservation states, i.e., the aggregated resource reservation window of a core node is K.

3. New resource reservation request succeed only if there are sufficient resources along the path. That is every related core node's remaining resource is no less than the amount of newly request resource. Otherwise, the resource reservation request failed. Resource is reserved in unit of time slot, and at most K

time slots. If a flow wants to consecutively reserve resources after the new resource reservation request expired, edge node/ host can send renewal request. Similar to new resource reservation request, renewal request also needs to carry the flow identifier (the same identifier as the flow identifier carried by the new resource reservation request), the amount of reserved resource (no more than the previous request), as well as the number of time slot that the resource will be reserved. Edge node/host also can active teardown the resource reservation along the path.

4. After receiving the the per-flow resource reservation message, core nodes refresh their aggregated resource reservation windows accordingly. As item 2 specifies, core node may record the sum of reserved_resource or the remaining resource (remaining resource = capacity - sum of reserved_resource). If the sum of reserved resources is recorded, then core node should add the newly requested resource to the maintained resource in each related time slot. Otherwise if the remaining resource is recorded, then core node should subtract the newly requested resource to the maintained resource in each related time slot.

5. Performance Analysis

5.1. Queueing Delay

We consider forwarding from an LDN node A via an LDN node B to an LDN node C and call the single-hop LDN delay the time between a packet being sent by A and the time it is re-sent by B. This single-hop delay is composed from the A->B propagation delay and the single-hop queuing delay A->B.

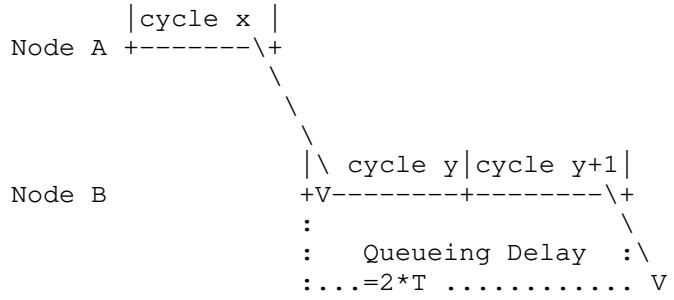


Figure 6: Single-Hop Queueing Delay

As Figure 6 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*T. Therefore the end-to-end queueing delay's upper bound is 2*T*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.

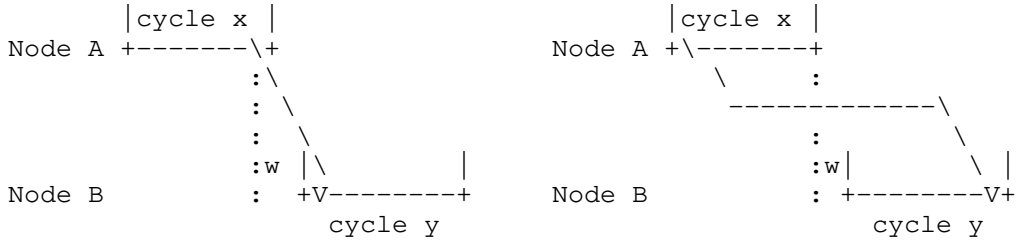
5.2. Jitter

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 7 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 + length of cycle x + length of cycle y= w+2*T;

- Hence the jitter's upper bound of this simplest scenario= worst case-best case=2*T.



(a) best situation (b) worst situation

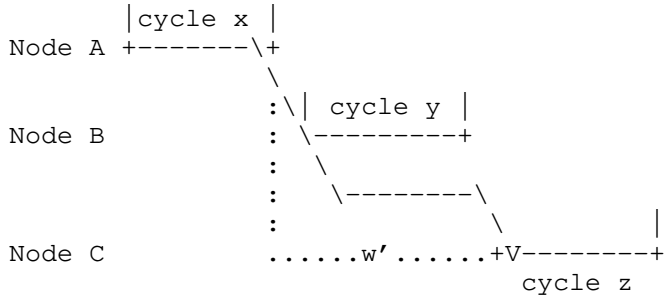
Figure 7: Jitter Analysis for One Hop Forwarding

Next considering two hops forwarding as Figure 8 shows.

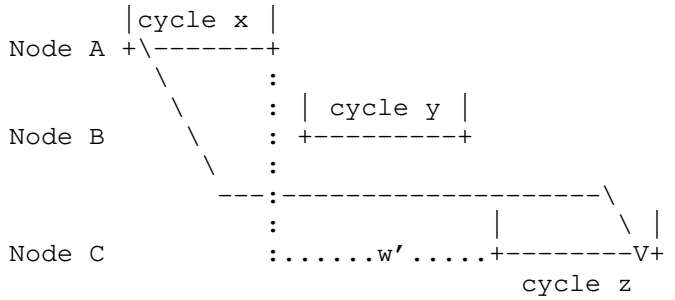
- The best situation is Node A sends packet at the end of cycle x, and Node C receives packet at the beginning of cycle z, then the delay is denoted by w';

- 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' + length of cycle x + 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 8: 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*T.

6. IANA Considerations

This document makes no request of IANA.

7. Security Considerations

Security issues have been carefully considered in [draft-ietf-detnet-security]. More discussion is TBD.

8. Acknowledgements

TBD.

9. 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>.

[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

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

Lei Wang
China Mobile
Beijing
China

Email: wangleiyjy@chinamobile.com

DeNet WG
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2019

Quan Xiong
Jinghai Yu
ZTE Corporation

October 18, 2018

DetNet QoS Policy
draft-xiong-detnet-qos-policy-00

Abstract

This document proposes a Quality of Service (QoS) policy to apply Differentiated Services (DiffServ) model for Deterministic Networking (DetNet) and defines a DetNet DiffServ mechanism including DetNet IP and MPLS encapsulation.

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 April 21, 2019.

Table of Contents

1. Introduction

As defined in [I-D.ietf-detnet-architecture], Deterministic Networking (DetNet) provides a capability to carry specified unicast or multicast data flows for real-time applications with extremely low data loss rates and bounded latency. DetNet and non-DetNet packets may be allowed to transmitted in the same network and more than one DetNet flows which has different priorities may be forwarded through the DetNet domain. The DetNet Class of Service (CoS) should be taken into consideration to provide Quality of Service (QoS) for DetNet services.

As discussed in [I-D.ietf-detnet-dp-sol-ip], Differentiated Services (DiffServ) can be used to provide traffic forwarding treatment for DetNet network. The DiffServ architecture as specified in [RFC2475] defined a model that traffic entering a DiffServ domain is classified and conditioned at the boundaries and marked with a DiffServ Code Point (DSCP) defined in [RFC2474]. The DSCP is used at transit nodes to select the Per Hop Behavior (PHB) that determines the scheduling treatment. And [RFC3270] provide a solution to support DiffServ for traffic marked with Traffic Class (TC) [RFC5462] transported over an MPLS network.

This document proposes a QoS policy to apply DiffServ model for DetNet network and defines a DetNet DiffServ mechanism including DetNet IP and MPLS encapsulation.

1.1. 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 [RFC2119].

1.2. Terminology

The terminology is defined as [I-D.ietf-detnet-architecture], [RFC3270], [RFC2475] and [RFC2474].

2. DetNet DiffServ Overview

The DetNet network needs to be capable of supporting differentiated services dividing to one or more contiguous DiffServ domains. The key components within a DiffServ domain including traffic classification and conditioning functions, and PHB-based forwarding. The customers may specify packet classification policy, traffic profiles and actions to DetNet flows which are in-profile or out-of-profile at the boundary. The DiffServ domains may support different PHB groups internally and different codepoint->PHB mappings at the transit nodes. The DetNet DiffServ process for packets is as Figure 1 shown.

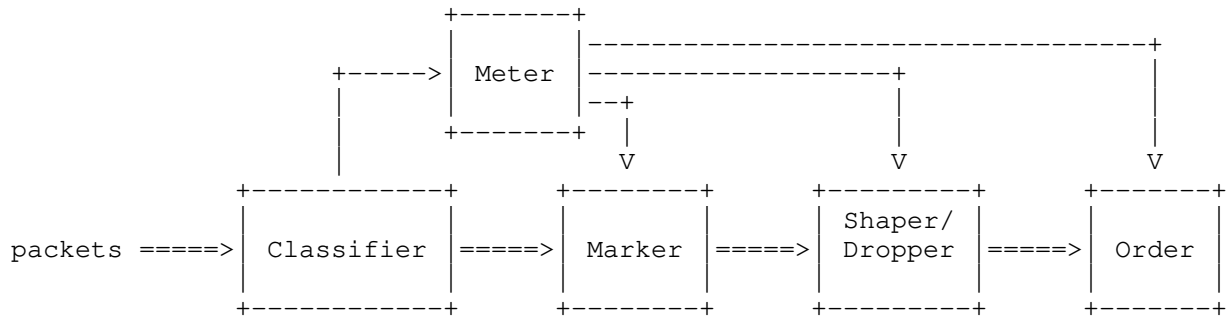


Figure 1: Overview of a DetNet DiffServ mechanism

2.1. DetNet Classifiers

As defined in [RFC2475], packet classifiers select packets in a traffic stream based on the information of packet header including two types of classifiers, the BA (Behavior Aggregate) and MF (Multi-Field) Classifier. The difference is that the BA classifies packets based on the CoS field and the latter one based on more other header fields.

In DetNet DiffServ model, BA and MF can be applied for packets classification. After classification, the flows can be seperated from DetNet and non-DetNet. As specified in [I-D.ietf-detnet-dp-sol-ip], no DetNet specific encapsulation is defined to support DetNet IP flow identification and DetNet service delivery. So the DetNet IP classifiers is the same as defined in [RFC2474] and [RFC2475]. As defined in [I-D.ietf-detnet-dp-sol-mpls], DetNet service Label (S-label) is defined to identify a DetNet flow in DetNet MPLS header. The S-label can be used in combination with MPLS TC filed in MF classifier. And DetNet MPLS BA classifier select packets based on the MPLS TC field only as defined in [RFC5462].

2.2. DetNet Traffic Conditioners

As [RFC2475] defined, the traffic conditioner may contain four elements: meter, marker, shaper and dropper. Traffic conditioning performs metering, shaping, policing and/or re-marking to ensure the traffic which entering the DiffServ domain conforms to the service provisioning policy.

A meter is used to measure the DetNet flows selected by a classifier and the result of the meter with respect to a particular packet may be used to trigger a particular action including a marking, dropping, or shaping. A marker is used to set the Cos field of a DetNet packet to a particular value, mapping the marked packet to a DetNet PHB. A Shaper may apply specific shaping algorithms implemented by DetNet network. A dropper is used to discard some of the non-DetNet packets to provide the QoS of the DetNet flows.

2.2.1. Order

As defined in [I-D.ietf-detnet-dp-sol-mpls], DetNet control word (d-CW) containing sequencing information for packet replication and duplicate elimination purposes. Sequence Number is different packet-by-packet. Based on Detnet MPLS date plane encapsulation, this document proposes a new type of action for DetNet traffic conditioning named order action which used to reorder the packets within a DetNet flow that are received out of order.

2.3. DetNet Per-hop Behavior

As specified in [RFC2475], per-hop behaviors are defined to permit a reasonably granular means of allocating buffer and bandwidth resources at each node among competing traffic streams. PHB groups will usually share a common constraint such as a packet scheduling or buffer management policy. According to [RFC4594], Default Forwarding (DF) PHB, Assured Forwarding (AF) PHBs, Expedited Forwarding (EF) PHB

and Class Selector (CS) PHBs have been defined to provide forwarding treatment. These PHBs can be used to forward DetNet flows based on the requirement.

This document defines a new Deterministic Networking (DN) PHB which is intended for traffic requiring extremely low data loss rates and bounded latency for DetNet. DetNet PHB specifications MUST be defined including a recommended default codepoint, which MUST be unique for codepoints in the standard space. The DSCP in IP header ans TC in MPLS header should be mapped to DN PHB with the relevant PHB specification which may be completed in future discussion.

2.4. DetNet Queuing

As discussed in [I-D.ietf-detnet-architecture],the nodes in DetNet network shall queue each received packets to one of the potential transmission ports and provide storage for queued packets, awaiting to submit these for transmission. A port provides one or more queues corresponding to the number of traffic classes. The queuing mechanism should be configured and implemented to DetNet nodes.

As defined in [RFC4594], Priority Queuing (PQ) was defined to queue the packets in priority sequence and Rate Queuing (RQ)selects packets according to the specified rate including Weighted Fair Queuing (WFQ) and Weighted Round Robin (WRR). Active Queue Management (AQM) also be defined to use packet dropping or marking to manage the depth of a queue.

As per IEEE 802.1 WG, queuing and transmission selection algorithms also can be used for queue scheduling in DetNet network.

3. DetNet IP DiffServ Consideration

As specified in [I-D.ietf-detnet-dp-sol-ip], no DetNet specific encapsulation is defined to support DetNet IP flow identification and DetNet service delivery. So the DetNet IP classification is the same as defined in [RFC2474] and [RFC2475]. But the recommended DetNet DSCP may be uesd to mark packets to select a DetNet PHB and the transit nodes should implement mechanisms performing the PHB. The mapping of DSCP to PHBs MUST be configurable. Implementations should support the recommended codepoint-to-PHB mappings in their default configuration.

4. DetNet MPLS DiffServ Consideration

As defined in [I-D.ietf-detnet-dp-sol-mpls], DetNet S-label is defined to identify a DetNet flow in DetNet MPLS header. The S-label

can be used in combination with MPLS TC filed in MF classifier. The BA classifier is the same with the [RFC3270].

Two types of LSPs including Explicitly TC-encoded-PSC LSP (E-LSP) and Label-Only-Inferred-PSC LSP (L-LSP) follows the definition of [RFC3270] and can be used to support DetNet explicit routes in MPLS-TE LSP. A E-LSP can be used to support one or more DetNet flows and a L-LSP can be established for one flow. E-LSP and L-LSP can use a signaled TC->PHB mapping to forward packets whose corresponding PHBs are defined in this document.

In DetNet MPLS network, DetNet Layer Two Service is supported in TSN over MPLS. The LSP egressing over egde nodes can use the preconfigured PHB->802.1 mapping as defined in [RFC3270].

As specified in [RFC3270], there may be more than one LSP carrying the same flow. Two or more LSPs can be merged into one LSP at one egressing LSR. It can be used to perform the packet replication (PRF) at ingress nodes and the packet elimination (PEF) at the egress nodes in DetNet DiffServ model. The order action which defined in this document can be used for packet ordering functionality (POF).

5. Security Considerations

TBD.

6. IANA Considerations

TBD.

7. Acknowledgements

TBD.

8. References

8.1. Informative References

[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>.

8.2. 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.

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

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

[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>.

[RFC4594] Babiarz, J., Chan, K., and F. Baker, "Configuration Guidelines for DiffServ Service Classes", RFC 4594, DOI 10.17487/RFC4594, August 2006, <https://www.rfc-editor.org/info/rfc4594>.

[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>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan, Hubei 430223
China

Phone: +86 27 83531060
Email: xiong.quan@zte.com.cn

Jinghai Yu
ZTE Corporation
50 Software Avenue, YuHuaTai District
Nanjing, Jiangsu 210012
China

Phone: +86 025 26774049
Email: yu.jinghai@zte.com.cn

DeNet WG
Internet-Draft
Intended status: Standards Track
Expires: April 22, 2019

Quan Xiong
Yufang Han
ZTE Corporation

October 19, 2018

DetNet QoS Yang
draft-xiong-detnet-qos-yang-00

Abstract

This document defines a YANG data model for Deterministic Networking (DetNet) Quality of Service (QoS) based on the Differentiated Services (DiffServ) model.

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 April 22, 2019.

Copyright Notice

Table of Contents

1. Introduction

Deterministic Networking (DetNet) as defined in [I-D.ietf-detnet-architecture], provides a capability to carry specified unicast or multicast data flows for real-time applications with extremely low data loss rates and bounded latency. In the meanwhile, DetNet and non-DetNet packets are allowed to be transmitted in the same network and more than one DetNet flows which has different priorities may be forwarded through the DetNet domain. As discussed in [I-D.ietf-detnet-dp-sol-ip] and [I-D.xiong-detnet-qos-policy], the Differentiated Services (DiffServ) can be used to provide Quality of Service (QoS) for DetNet services.

This document defines a YANG data model for DetNet QoS based on the DiffServ model.

1.1. 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 [RFC2119].

1.2. Terminology

The terminology is defined as [I-D.ietf-detnet-architecture], [RFC3270], [RFC2474] and [RFC2475].

2. DetNet DiffServ QoS Model

This document defines a YANG data module for DetNet DiffServ QoS Model as discussed in [I-D.xiong-detnet-qos-policy]. In the ietf-detnet-qos module, this is performed as one of the DetNet QoS policy.

2.1. DetNet QoS Tree Structure

DetNet DiffServ model is one type of the DetNet QoS policy and other policy types can be defined in detnet-policy-type.

[I-D.xiong-detnet-qos-policy] specified two types of classifiers including BA (Behavior Aggregate) and MF (Multi-Field) classifiers in detnet-classifier-type. DetNet IP BA classifier selects packets based on the DiffServ Code Point (DSCP) and DetNet MPLS BA classifier is based on the MPLS Traffic Class (TC) field. DetNet IP MF classifier selects packets based on the value of a combination of source address, destination address, DSCP, protocol ID, source port and destination port numbers and DetNet MPLS MF classifier is based on the MPLS TC and Service Label (S-Label) field of the header.

[I-D.xiong-detnet-qos-policy] defined a DetNet (DN) Per Hop Behavior (PHB) for DetNet forward other than existing PHBs including AF,EF,CS,DF etc. The PHB class information description is as qos-phb-class shown.

[I-D.xiong-detnet-qos-policy] defined a new type of action for DetNet traffic conditioning named order action to reorder the packets. Other actions including meter, shaper, dropper and marker as the detnet-action-type shown.

```
module: ietf-detnet-qos
    +--rw detnet-qos-policies
       +--rw detnet-policy-template* [detnet-policy-name]
          +--rw detnet-policy-name            string
          +--rw detnet-policy-type?           detnet-policy-type
          +--rw detnet-classifier-template* [detnet-classifier-name]
             +--rw detnet-classifier-name    string
             +--rw detnet-classifier-type?   detnet-classifier-type
             +--rw (classifier-type)?
             |  +--:(ba)
             |  |  +--rw (encapsulation-type)?
             |  |     +--:(MPLS)
             |  |     |  +--rw mpls-ba* [tc-value]
             |  |     |     +--rw phb-class?   qos-phb-class
             |  |     |     +--rw tc-value     uint8
             |  |     +--:(IP)
             |  |        +--rw ip-ba* [dscp-value]
             |  |           +--rw phb-class?    qos-phb-class
             |  |           +--rw dscp-value    uint8
             |  +--:(mf)
             |     +--rw (tunnel-type)?
             |        +--:(MPLS)
             |        |  +--rw mpls-mf* [tc-value]
```

```
               |         |     +--rw phb-class?   qos-phb-class
               |         |     +--rw tc-value     uint8
               |         |     +--rw s-label?     uint32
               |         +--:(IPv4)
               |         |  +--rw ipv4-mf* [dscp-value]
               |         |     +--rw phb-class?                  qos-phb-class
               |         |     +--rw dscp-value                  uint8
               |         |     +--rw ipv4-source-address?        inet:ipv4-address
               |         |     +--rw ipv4-destination-address?   inet:ipv4-address
               |         |     +--rw protocol-ID?                uint8
               |         |     +--rw source-port-numbers?        inet:port-number
               |         |     +--rw destination-port-numbers?   inet:port-number
               |         +--:(IPv6)
               |            +--rw ipv6-mf* [dscp-value]
               |               +--rw phb-class?                  qos-phb-class
               |               +--rw dscp-value                  uint8
               |               +--rw ipv6-source-address?        inet:ipv6-address
               |               +--rw ipv6-destination-address?   inet:ipv6-address
               |               +--rw protocol-ID?                uint8
               |               +--rw source-port-numbers?        inet:port-number
               |               +--rw destination-port-numbers?   inet:port-number
               +--rw detnet-action* [detnet-action-type]
                  +--rw detnet-action-type    detnet-action-type
                  +--rw (actions)?
                     +--:(meter)
                     +--:(marker)
                     +--:(shaper)
                     +--:(dropper)
                     +--:(order)
```

2.2. DetNet QoS Module

```
<CODE BEGINS> file "detnet-diffserv-qos@2018-10-13.yang"
module ietf-detnet-qos {
     yang-version 1;
     namespace "urn:ietf:params:xml:ns:yang:ietf-detnet-qos";
     prefix detnet-qos;

         import ietf-inet-types{
       prefix "inet";
     }

     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:   Quan Xiong
                  <mailto:xiong.quan@zte.com.cn>
        Editor:   Yufang Han
                  <mailto:han.yufang1@zte.com.cn>";

     description
       "This YANG module describes the Deterministic Networking (DetNet)
           Quality of Service (QoS) based on the Differentiated Services (DiffSe
rv)
           model.";

     revision "2018-10-13" {
       description "initial revision";
       reference "RFC XXXX: draft-xiong-detnet-qos-yang-00";
     }

    typedef qos-phb-class {
     type enumeration {
      enum df {
        value 1 ;
        description "Default Forwarding for Best effort";
      }
      enum af1 {
        value 2 ;
        description "Assured forwarding class 1";
      }
      enum af2 {
        value 3 ;
        description "Assured forwarding class 2";
      }
      enum af3 {
        value 4 ;
        description "Assured forwarding class 3";
      }
      enum af4 {
        value 5 ;
        description "Assured forwarding class 4";
      }
      enum ef {
        value 6 ;
        description "Expedited forward";
      }
          enum cs6 {
        value 7 ;
        description "Internetwork control service class";
      }
      enum cs7 {
```

```
        value 8 ;
        description "Network control service class";
      }
      enum dn {
        value 9 ;
        description "DetNet forward";
      }
     }
         description
         "The PHB class including AF,EF,CS,DF,DN";
    }

    typedef detnet-policy-type {
          type enumeration {
            enum diffserv {
              value 1 ;
                  description "DiffServ Policy";
            }
          }
          description
         "The DetNet policy type.";
    }

    typedef detnet-classifier-type {
          type enumeration {
            enum ba {
              value 1 ;
                  description "DiffServ BA Classifier";
            }
                enum mf {
              value 2 ;
                  description "DiffServ MF Classifier";
            }
          }
          description
         "The DetNet classifier type including BA and MF.";
    }

        typedef detnet-action-type {
          type enumeration {
            enum meter {
              value 1 ;
                  description "DiffServ meter Action";
            }
            enum shaper {
              value 2 ;
                  description "DiffServ shaper Action";
            }
```

```
                enum dropper {
              value 3 ;
                  description "DiffServ dropper Action";
            }
                enum marker {
              value 4 ;
                  description "DiffServ marker Action";
            }
                enum order {
              value 5 ;
                  description "DiffServ order Action";
            }
          }
          description
           "The DetNet classifier type including BA and MF.";
        }

        grouping mpls-tc {
        description "MPLS TC Information";
        leaf phb-class {
                    type qos-phb-class;
            description "Specify phb class of PHB info, support [a"
            + "f1,af2,af3,af4,be,ef,cs6,cs7,dt]";
        }
        leaf tc-value {
            type uint8 {
                range 0..7 {
                description "MPLS-TC value, support [0-7]";
                }
            }
            mandatory true ;
                        description "MPLS-TC value, support [0-7]";
            }
    }

        grouping ip-dscp {
        description "IP DSCP Information";
        leaf phb-class {
                    type qos-phb-class ;
            description "Specify server class of PHB info, support [a"
            + "f1,af2,af3,af4,be,ef,cs6,cs7,dt]";
        }
        leaf dscp-value {
            type uint8 {
                range 0..63 {
                description "IPv4/IPv6 DSCP value, support [0-63]";
                }
            }
```

```
            mandatory true ;
                        description "IPv4/IPv6 DSCP value, support [0-63]";
            }
    }

    grouping mpls-header-info {
        description "MPLS TC Information";
        leaf phb-class {
                    type qos-phb-class ;
            description "Specify phb class of PHB info, support [a"
            + "f1,af2,af3,af4,be,ef,cs6,cs7,dt]";
        }
        leaf tc-value {
            type uint8 {
                range 0..7 {
                description "MPLS-TC value, support [0-7]";
                }
            }
            mandatory true ;
                        description "MPLS-TC value, support [0-7]";
            }
                leaf s-label {
                    type uint32;
                    description "DetNet Flow ID value, support classifier MF";
                }
    }

        grouping ipv4-header-info {
        description "IP DSCP Information";
        leaf phb-class {
                    type qos-phb-class ;
            description "Specify server class of PHB info, support [a"
            + "f1,af2,af3,af4,be,ef,cs6,cs7,dt]";
        }
        leaf dscp-value {
            type uint8 {
                range 0..63 {
                description "IPv4/IPv6 DSCP value, support [0-63]";
                }
            }
            mandatory true ;
                        description "IPv4/IPv6 DSCP value, support [0-63]";
            }
                leaf ipv4-source-address {
                        type inet:ipv4-address;
                    description "source address value, support classifier MF";
                }
                leaf ipv4-destination-address {
```

```
                        type inet:ipv4-address;
                    description "destination address value, support classifier M
F";
                }
                leaf protocol-ID {
                    type uint8;
                    description "protocol ID, support classifier MF";
                }
                leaf source-port-numbers {
                    type inet:port-number;
                    description "source port numbers, support classifier MF";
                }
                leaf destination-port-numbers {
                    type inet:port-number;
                    description "destination port numbers, support classifier MF
";
                }
    }

        grouping ipv6-header-info {
        description "IPv6 DSCP Information";
        leaf phb-class {
                    type qos-phb-class ;
            description "Specify server class of PHB info, support [a"
            + "f1,af2,af3,af4,be,ef,cs6,cs7,dt]";
        }
        leaf dscp-value {
            type uint8 {
                range 0..63 {
                description "IPv4/IPv6 DSCP value, support [0-63]";
                }
            }
            mandatory true ;
                        description "IPv4/IPv6 DSCP value, support [0-63]";
            }
                leaf ipv6-source-address {
                        type inet:ipv6-address;
                    description "source address value, support classifier MF";
                }
                leaf ipv6-destination-address {
                        type inet:ipv6-address;
                    description "destination address value, support classifier M
F";
                }
                leaf protocol-ID {
                    type uint8;
                    description "protocol ID, support classifier MF";
                }
                leaf source-port-numbers {
                    type inet:port-number;
                    description "source port numbers, support classifier MF";
```

```
                }
                leaf destination-port-numbers {
                    type inet:port-number;
                    description "destination port numbers, support classifier MF
";
                }
    }

    grouping detnet-classifiers {
        description "Configure the DetNet classifiers";
        choice classifier-type {
            description "Choice of classifiers types";
                        case ba {
                            description "BA classifier";
                                choice encapsulation-type {
                  description "Tunnel type includes: IP, MPLS.";
                  case MPLS {
                                    list mpls-ba {
                        key "tc-value";
                                            description "MPLS-TC be mapped to PH
B";
                        uses mpls-tc;
                    }
                  }
                  case IP {
                    list ip-ba {
                        key "dscp-value";
                                            description "IPv4/IPv6 DSCP be mappe
d to PHB";
                        uses ip-dscp;
                                    }
                  }
                }
                        }
                        case mf {
                            description "MF classifier";
                                choice tunnel-type {
                  description
                     "Tunnel type includes: IPv4, IPv6, MPLS.";
                  case MPLS {
                                    list mpls-mf {
                        key "tc-value";
                                            description "MPLS-TC be mapped to PH
B";
                        uses mpls-header-info;
                    }
                  }
                  case IPv4 {
                    list ipv4-mf {
                        key "dscp-value";
                                            description "IPv4 DSCP be mapped to
PHB";
                        uses ipv4-header-info;
```

```
                                        }
                                    }
                  case IPv6 {
                    list ipv6-mf {
                        key "dscp-value";
                                        description "IPv6 DSCP be mapped to
PHB";
                        uses ipv6-header-info;
                                        }
                                    }
                }
                        }
        }
    }

        grouping detnet-actions {
        description
          "DetNet Configuration about the actions";
        list detnet-action {
            key "detnet-action-type";
                        description "DetNet actions, to be defined.";
            leaf detnet-action-type {
              type detnet-action-type;
                          description "DetNet action types";
            }
            choice actions {
              description "Choice of action types";
                          case meter {
                            description "meter action";
              }
                          case marker {
                            description "marker action";
              }
                          case shaper {
                            description "shaper action";
              }
                          case dropper {
                            description "dropper action";
              }
                          case order {
                            description "order action";
              }
            }
        }
    }

        container detnet-qos-policies {
        description "Configuration about DetNet QoS Policy";
        list detnet-policy-template {
```

```
                        key detnet-policy-name;
                        description "DetNet policy template";
                    leaf detnet-policy-name {
                           type string;
                           description "DetNet policy name";
            }
            leaf detnet-policy-type {
              type detnet-policy-type;
                          description "DetNet policy type";
            }
            list detnet-classifier-template {
                key detnet-classifier-name;
                                description "DetNet classifier template";
                leaf detnet-classifier-name {
                    type string;
                                        description "DetNet classifier name";
                }
                                leaf detnet-classifier-type {
                                        type detnet-classifier-type;
                                        description "DetNet classifier type";
                }
                        uses detnet-classifiers;
                                uses detnet-actions;
            }
            }
        }
}
```

<CODE ENDS>

3. Security Considerations

TBD.

4. IANA Considerations

TBD.

5. Acknowledgements

TBD.

6. References

6.1. Informative References

[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>.

6.2. 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.xiong-detnet-qos-policy] Xiong, Q. and Y. jinghai, "DetNet QoS Policy", draft-xiong-detnet-qos-policy-00 (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>.

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

[RFC3270] Le Faucheur, F., Wu, L., Davie, B., Davari, S., Vaananen, P., Krishnan, R., Cheval, P., and J. Heinanen, "Multi-Protocol Label Switching (MPLS) Support of Differentiated Services", RFC 3270, DOI 10.17487/RFC3270, May 2002, <https://www.rfc-editor.org/info/rfc3270>.

[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>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan, Hubei 430223
China

Phone: +86 27 83531060
Email: xiong.quan@zte.com.cn

Yufang Han
ZTE Corporation
50 Software Avenue, YuHuaTai District
Nanjing, Jiangsu 210012
China

Phone: +86 15951984307
Email: han.yufang1@zte.com.cn