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

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

October 22, 2018

DetNet Bounded Latency
draft-finn-detnet-bounded-latency-02

Abstract

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

Status of This Memo

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

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

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

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The ability for IETF Deterministic Networking (DetNet) or IEEE 802.1 Time-Sensitive Networking (TSN, [IEEE8021TSN]) to provide the DetNet services of bounded latency and zero congestion loss depends upon A)

configuring and allocating network resources for the exclusive use of DetNet/TSN flows; B) identifying, in the data plane, the resources to be utilized by any given packet, and C) the detailed behavior of those resources, especially transmission queue selection, so that latency bounds can be reliably assured. Thus, DetNet is an example of an INTSERV Guaranteed Quality of Service [RFC2212]

As explained in [I-D.ietf-detnet-architecture], DetNet flows are characterized by 1) a maximum bandwidth, guaranteed either by the transmitter or by strict input metering; and 2) a requirement for a guaranteed worst-case end-to-end latency. That latency guarantee, in turn, provides the opportunity for the network to supply enough buffer space to guarantee zero congestion loss. To be of use to the applications identified in [I-D.ietf-detnet-use-cases], it must be possible to calculate, before the transmission of a DetNet flow commences, both the worst-case end-to-end network latency, and the amount of buffer space required at each hop to ensure against congestion loss.

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

- Characterize a DetNet flow in a way that provides externally measurable verification that the sender is conforming to its promised maximum, can be implemented reasonably easily by a sending device, and does not require excessive over-allocation of resources by the network.

- Enable reasonably accurate computation of worst-case end-to-end latency, in a way that requires as little detailed knowledge as possible of the behavior of the Quality of Service (QoS)

algorithms implemented in each device, including queuing, shaping, metering, policing, and transmission selection techniques.

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

This document does not specify any resource reservation protocol or server. It does not describe all of the requirements for that

protocol or server. It does describe requirements for such resource reservation methods, and for queuing mechanisms that, if met, will enable them to work together.

NOTE: This draft is not yet complete, but it is sufficiently so to share with the Working Group and to obtain opinions and direction. The present intent of is for this draft to become a normative RFC, defining how one SHALL/SHOULD provide the DetNet quality of service. There are still a few authors' notes to each other present in this draft.

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 assumes the use of the following paradigm for provisioning a particular DetNet flow:

1. Perform any configuration 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 not tied to any particular flow.

2. Characterize the DetNet flow in terms of limitations on the sender [Section 9.1] and flow requirements [Section 9.2].

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 to provide zero congestion loss.

6. Assuming that the resources are available, commit those resources to the flow. This may or may not require adjusting the parameters that control the queuing mechanisms at each hop along the flow's path.

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

Issues such as un-provisioning a DetNet flow in favor of another when resources are scarce are not considered. How the path to be taken by a DetNet flow is chosen is not considered in this document.

4.2. Relay system model

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.

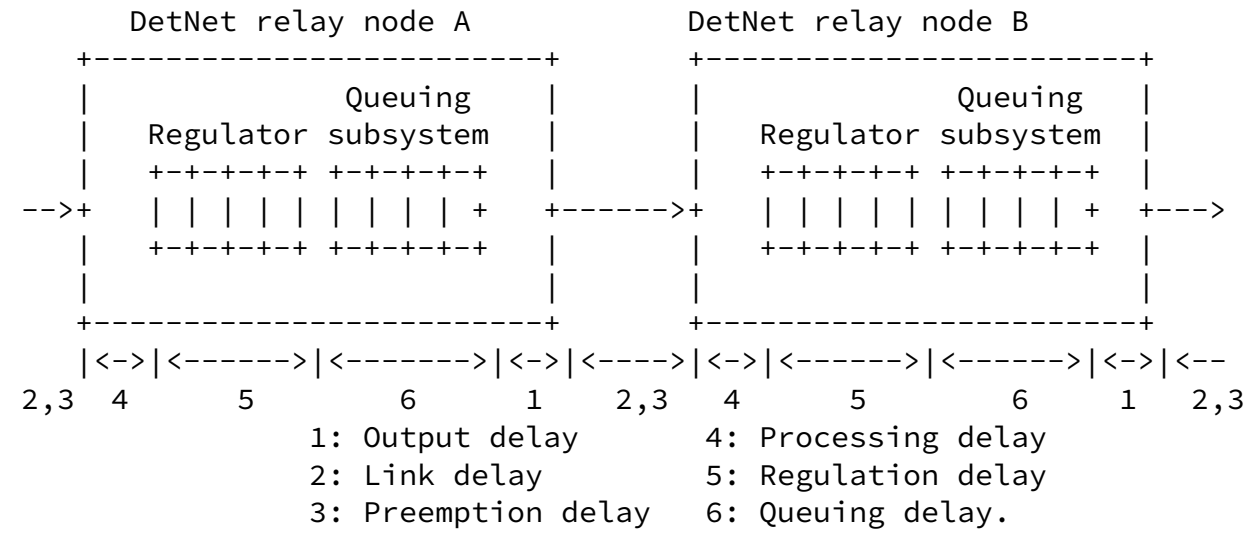

Figure 1: Timing model for DetNet or TSN

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 six delays that a packet can experience from hop to hop.

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

2. Link delay
   The time taken from the transmission of the first bit of the

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

3. Preemption delay
If the packet is interrupted (e.g. [IEEE8023br] and [IEEE8021Qbu] 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 the time the packet is enqueued in the regulator (Queuing subsystem, if there is no regulation). This delay can be variable, and depends on the details of the operation of the forwarding node.

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

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

insertion of the packet into a queue until it is selected for output on the next link.

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

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

- 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 subsystem delay (6) calculations, these variations need not be visible outside the DetNet relay node.

5. Computing End-to-end Latency Bounds

5.1. Non-queuing delay bound

End-to-end latency bounds can be computed using the delay model in Section 4.2. 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.

5.2. Queuing delay bound

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. This occurs with (1) per-flow queuing, and (2) per-class queuing with regulators, as explained in Section 5.2.1, Section 5.2.2, and Section 7.

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.

5.2.1. Per-flow queuing mechanisms

With such mechanisms, each flow uses a separate queue inside every node. The service for each queue is abstracted with a guaranteed rate and a delay. For every flow the per-node delay bound as well as end-to-end delay bound can be computed from the traffic specification of this flow at its source and from the values of rates and latencies at all nodes along its path. Details of calculation for IntServ are described in Section 7.5.

5.2.2. Per-class queuing mechanisms

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

A solution to deal with this issue is to reshape the flows at every hop. This can be done with per-flow regulators (e.g. leaky bucket shapers), but this requires per-flow queuing and defeats the purpose of per-class queuing. An alternative is the interleaved regulator, which reshapes individual flows without per-flow queuing ([Specht2016UBS], [IEEE8021Qcr]"). With an interleaved regulator, the packet at the head of the queue is regulated based on its (flow) regulation constraints; it is released at the earliest time at which this is possible without violating the constraint. One key feature of per-flow or interleaved regulator is that, it does not increase worst-case latency bounds [le_boudec_theory_2018]. Specifically, when an interleaved regulator is appended to a FIFO subsystem, it does not increase the worst-case delay of the latter.

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

```
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 queuing subsystem in node i and interleaved regulator of node j, and S4 is a bound on the response time of the queuing subsystem in node 4 for flow f. In fact, using the delay definitions in Section 4.2, 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. A practical example of queuing model and delay calculation is presented Section 7.4.

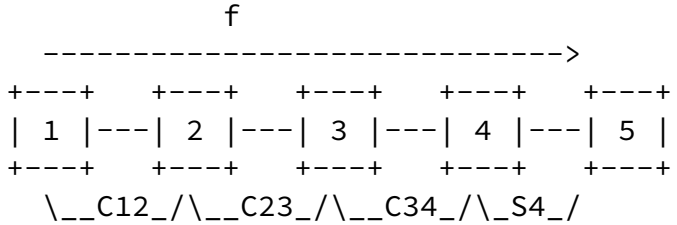

Figure 2: End-to-end latency computation example

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

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

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

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

Let

- 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.

- 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

7.1. Queuing data model

Sophisticated queuing mechanisms are available in Layer 3 (L3, see, e.g., [RFC7806] for an overview). In general, we assume that "Layer 3" queues, shapers, meters, etc., are precisely the "regulators" shown in Figure 1. The "queuing subsystems" in this figure are not the province solely of bridges; they are an essential part of any DetNet relay node. As illustrated by numerous implementation examples, some of the "Layer 3" mechanisms described in documents such as [RFC7806] are often integrated, in an implementation, with the "Layer 2" mechanisms also implemented in the same 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 general model for the flow of packets through the queues of a DetNet relay node. Packets are assigned to a class of service. The classes of service are mapped to some number of regulator queues. Only DetNet/TSN packets pass through regulators. Queues compete for the selection of packets to be passed to queues in the queuing subsystem. Packets again are selected for output from the queuing subsystem.

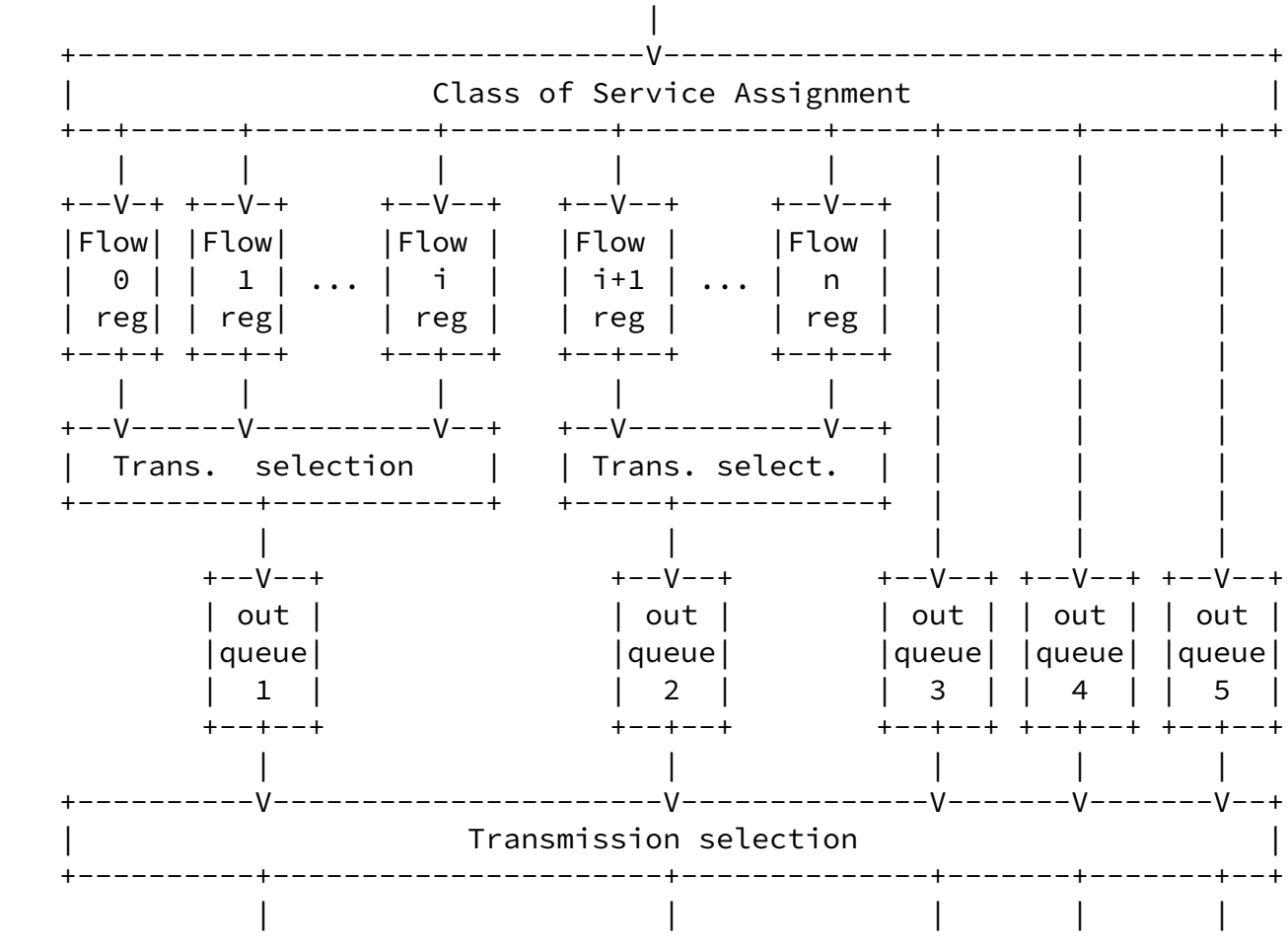
Class of Service Assignment
Flow 0 reg
Flow 1 reg
Flow i reg
Flow i+1 reg
Flow n reg
Trans. selection
Trans. select.
out queue 1
out queue 2
out queue 3
out queue 4
out queue 5
Transmission selection

Figure 3: IEEE 802.1Q Queuing Model: Data flow

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

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

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

The Class of Service Assignment function can be quite complex, even in a bridge [IEEE8021Q], since the introduction of [IEEE802.1Qci]. In addition to the Layer 2 priority expressed in the 802.1Q VLAN tag, a DetNet relay node 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. ([I-D.ietf-detnet-dp-sol-ip], [I-D.ietf-detnet-dp-sol-mpls]) (Work items are expected to add MPC and other indicators.)
- The Class of Service Assignment function can contain metering and policing functions.
- MPLS and/or pseudowire ([RFC6658]) labels.

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

7.2. Preemption

In IEEE Std 802.1Q, preemption is modeled as consisting of two MAC/PHY stacks, one for packets that can be interrupted, and one for packets that can interrupt the interruptible packets. The Class of Service (queue) determines which packets are which. Only one layer of preemption is supported. DetNet flows pass through the interrupting MAC. Only best-effort queues pass through the interruptible MAC, and can thus be preempted.

7.3. Time-scheduled queuing

In [IEEE8021Qbv], the notion of time-scheduling queue gates were introduced. On below every output queue (the lower row of queues in Figure 3) is a gate that permits or denies the queue to present data for transmission selection. The gates are controlled by a rotating schedule that can be locked to a clock that is synchronized with other relay nodes. The DetNet class of service can be supplied by queuing mechanisms based on time, rather than the regulator model in Figure 3. These queuing mechanisms are discussed in Section 8, below.

7.4. 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 4. The packets received at a node input port for a given class are enqueued in the respective interleaved regulator at the output port. Then, the packets from all the flows, including CDT and BE flows, are enqueued in a class based FIFO system (CBFS) [TSNwithATS].

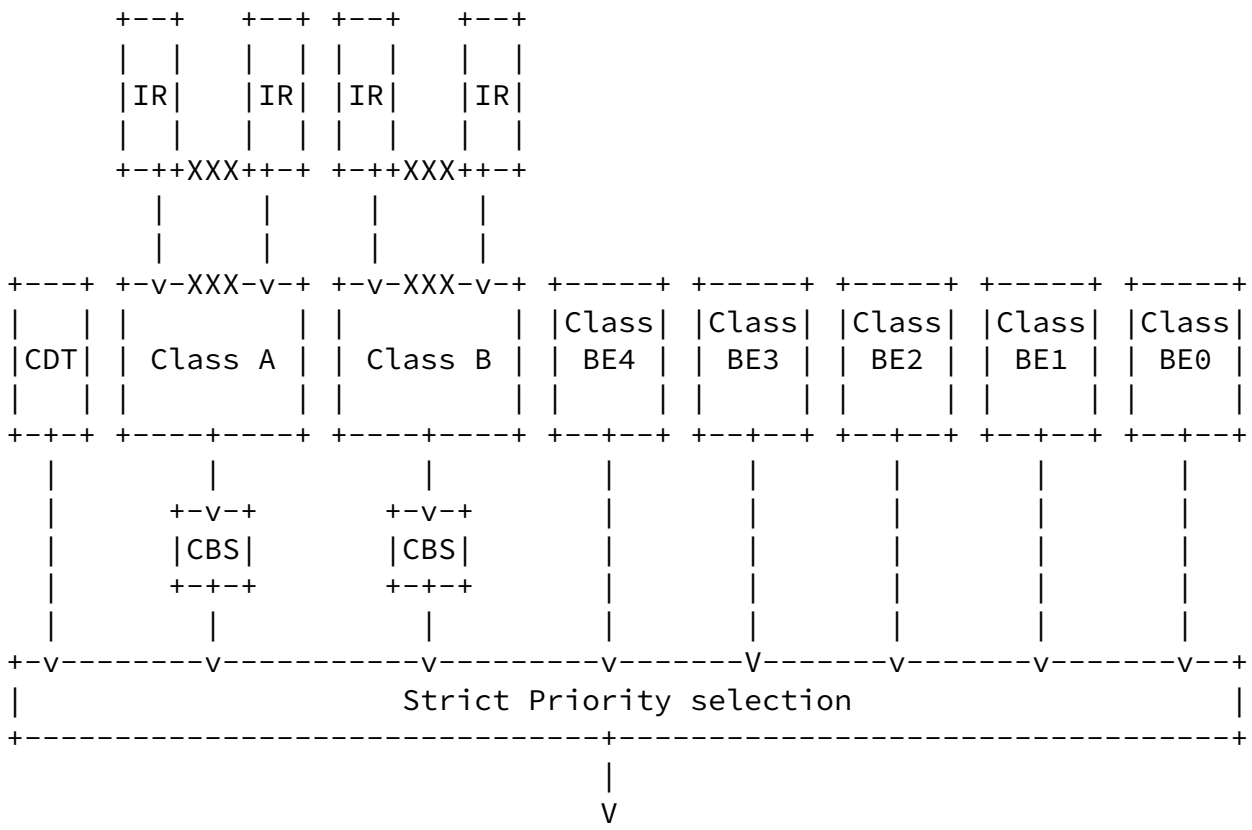

Figure 4: Architecture of a TSN node output port with interleaved regulators (IRs)

The CBFS includes two 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 9.2.]]

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.

Details of end-to-end delay bound calculation in such a system is described in [TSNwithATS].

7.5. IntServ

In this section, a worst-case queuing latency calculating method is provided. In deterministic network, the traffic of a flow is constrained by arrival curve. Queuing mechanisms in a DetNet node can be characterized and constrained by service curve. By using arrival curve and service curve with Network Calculus theory [NetCalBook], a tight worst-case queuing latency can be calculated.

Considering a DetNet flow at output port, R(s) is the cumulative arrival data until time s. For any time period t, the incremental arrival data is constrained by an arrival curve a(t)

$$R(s+t)-R(s) <= a(t), \any s>=0, t>=0$$

The scheduling that a relay node performs to a DetNet flow can be abstracted as service curve. It describes the minimal service the network can offer. The service curve b(t) of a node is defined as below, if the accumulative input data R and output data R_out of the node satisfies

R_out(t) >= inf(R(s) + b(t-s)), \any s <=t

where the operator "inf" calculates the greatest lower bound in period t.

By calculating the maximum vertical deviation between arrival curve a(t) and service curve b(t), one can obtain the backlog bound in data unit

Backlog_bound = sup_t(a(t) - b(t))

where operator "sup_t" calculates the minimum upper bound with respect to t. The buffer space at a node should be no less than the backlog bound to achieve zero congestion loss.

NOTE: Section 6.1 gives a general formula for computing the buffer requirements. This is an alternative calculation based on the arrival curve and service curve.

By calculating the maximum horizontal deviation between arrival curve a(t) and service curve b(t), one can obtain the delay bound as below

Delay_bound = sup_s(inf_t(t>=0 | a(s) <= b(s+t))

where the operator " inf_t" calculates the maximum lower bound with respect to t, the operator "sup_s" calculates the minimum upper bound with respect to s. Figure 5 shows an example of arival curve, service curve, backlog bound h, and delay bound v.

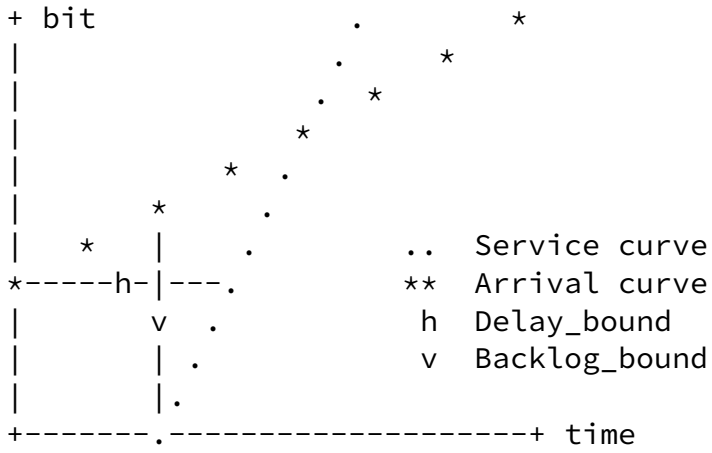

Figure 5: Computation of backlog bound and delay bound. Note that arrival and service curves are not necessary to be linear.

Note that in the formula of Delay_bound, the service curve b(t) can describe either per-hop scheduling that a DetNet node offers to a flow, or concatenation of multiple nodes that represents end-to-end scheduling that DetNet path offers to a flow. In the latter case, the obtained delay bound is end-to-end worst case delay. To calculate this, we should at first derive the concatenated service curve.

Consider a flow traverse two DetNet nodes, which offer service curve b1(t) and b2(t) sequentially. Then concatenation of the two nodes offers a service curve b_concatenated as below

b_concatenated(t) =inf_s (b1(s) + b2(t-s)) , \any 0 <=s <=t

The concatenation of service curve can be directly generalized to include more than two nodes.

a_out(t) = sup_u(a(t+u) - b(u)), \any u>=0

In DetNet, the arrival curve and service curve can be characterized by a group of parameters, which will be defined in Section 8.

Integrated service (IntServ) is an architecture that specifies the elements to guarantee quality of service (QoS) on networks. To

satisfied guaranteed service, a flow must conform to a traffic specification (T-spec), and reservation is made along a path, only if routers are able to guarantee the required bandwidth and buffer.

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

o Token bucket depth (b).

o Token bucket rate (r).

The traffic specification can be described as an arrival curve a(t)

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

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

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

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

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

It should be noted that, the guaranteed service rate R is a share of link's bandwidth. The choice of R is related to the specification of flows which will transmit on this node. For example, in strict priority policy, considering a flow with priority j, its share of bandwidth may be R=c-sum(r_i), i<j, where c is the link bandwidth, r_i is the token bucket rate for the flows with priority higher than j. The choice of T is also related to the specification of all the flows traversing this node. For example, in a generalized processor sharing (GPS) node, T = L / R + L_max/c, where L is the maximum packet size for the flow, L_max is the maximum packet size in the node across all flows. Other choice of R and T are also supported, according to the specific scheduling of the node and flows traversing this node.

As mentioned previously in this section, delay bound and backlog

bound can be easily obtained by comparing arrival curve and service curve. Backlog bound, or buffer bound, is the maximum vertical derivation between curves alpha(t) and beta(t), which is x=b+rT. Delay bound is the maximum horizontal derivation between curves alpha(t) and beta(t), which is d = T+b/R. Graphical illustration of the IntServ model is shown in Figure 5.

The output bound, or the next-hop arrival curve, is alpha_out(t) = b + rT + rt, where burstiness of the flow is increased by rT, compared with the arrival curve.

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

beta_i(t) = max(0, R_i(t-T_i)), i=1,...,N

According to [Section 5.1], by concatenating those IntServ nodes, an end-to-end service curve can be computed as

beta_e2e (t) = max(0, R_e2e(t-T_e2e))

where

R_e2e = min(R_1,..., R_N)

T_e2e = T_1 + ... + T_N

Similarly, delay bound, backlog bound and output bound can be computed by using the original arrival curve alpha(t) and concatenated service curve beta_e2e(t).

8. Time-based DetNet QoS

8.1. Cyclic Queuing and Forwarding

[IEEE802.1Qci] and [IEEE802.1Qch] describe Cyclic Queuing and Forwarding (CQF), which provide the bounded latency and zero congestion loss using the time-scheduled gates of [IEEE8021Qbv]. For each different DetNet class of service, a set of two or three buffers is provided at the out queue layer of Figure 3. A cycle time is configured for each class, and all of the buffer sets in a class swap buffers simultaneously throughout the DetNet domain at that cycle

rate. The choice of using two or three buffers depends on the link lengths and forwarding delay times; two buffers can be used if the delay from hop to hop is nearly an integral number of cycle times, and three are required if not. Flows are assigned to a class of service only until the amount of data to be transmitted in one cycle would exceed the cycle time for some interface. Every packet dwells either two or three cycles at each hop, so the calculation of worst-case latency and latency variation is trivial.

8.2. Time Scheduled Queuing

[IEEE8021Qbv] specifies a time-aware queue-draining procedure for transmission selection at egress port of a relay node, which supports up to eight traffic classes. Each traffic class has a separate queue, frame transmission from each queue is allowed or prevented by a time gate. This time gate controlled scheduling allows time-sensitive traffic classes to transmit on dedicate time slots. Within the time slots, the transmitting flows can be granted exclusive use of the transmission medium. Generally, this time-aware scheduling is a layer 2 time division multiplexing (TDM) technique.

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

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

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

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

9. Parameters for the bounded latency model

9.1. Sender parameters

9.2. Relay system parameters

[[NWF This section talks about the parameters that must be used hop-by-hop by a resource reservation protocol.]]

10. References

10.1. Normative References

[I-D.ietf-detnet-architecture]
Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", draft-ietf-

detnet-architecture-08 (work in progress), September 2018.

[I-D.ietf-detnet-dp-sol-ip]
Korhonen, J. and B. Varga, "DetNet IP Data Plane Encapsulation", draft-ietf-detnet-dp-sol-ip-00 (work in progress), July 2018.

[I-D.ietf-detnet-dp-sol-mpls]
Korhonen, J. and B. Varga, "DetNet MPLS Data Plane Encapsulation", draft-ietf-detnet-dp-sol-mpls-00 (work in progress), July 2018.

[I-D.ietf-detnet-use-cases]
Grossman, E., "Deterministic Networking Use Cases", draft-ietf-detnet-use-cases-19 (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>.

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

[RFC6658] Bryant, S., Ed., Martini, L., Swallow, G., and A. Malis, "Packet Pseudowire Encapsulation over an MPLS PSN", RFC 6658, DOI 10.17487/RFC6658, July 2012, <https://www.rfc-editor.org/info/rfc6658>.

[RFC7806] Baker, F. and R. Pan, "On Queuing, Marking, and Dropping", RFC 7806, DOI 10.17487/RFC7806, April 2016, <https://www.rfc-editor.org/info/rfc7806>.

10.2. Informative References

[bennett2002delay]
J.C.R. Bennett, K. Benson, A. Charny, W.F. Courtney, and J.-Y. Le Boudec, "Delay Jitter Bounds and Packet Scale

Rate Guarantee for Expedited Forwarding", <https://dl.acm.org/citation.cfm?id=581870>.

[charny2000delay]
A. Charny and J.-Y. Le Boudec, "Delay Bounds in a Network with Aggregate Scheduling", <https://link.springer.com/chapter/10.1007/3-540-39939-9_1>.

[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/>.

[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>.

[le_boudec_theory_2018]
J.-Y. Le Boudec, "A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators", <http://arxiv.org/abs/1801.08477/>.

[NetCalBook]
Le Boudec, Jean-Yves, and Patrick Thiran, "Network calculus: a theory of deterministic queuing systems for the internet", 2001, <https://arxiv.org/abs/1804.10608/>.

[Specht2016UBS]
J. Specht and S. Samii, "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks", <https://ieeexplore.ieee.org/abstract/document/7557870>.

[TSNwithATS]
E. Mohammadpour, E. Stai, M. Mohiuddin, and J.-Y. Le Boudec, "End-to-end Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping", <https://arxiv.org/abs/1804.10608/>.

Authors' Addresses

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

Phone: +1 925 980 6430
Email: norman.finn@mail01.huawei.com

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

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

Email: zhangjiayi11@huawei.com

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