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.

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.

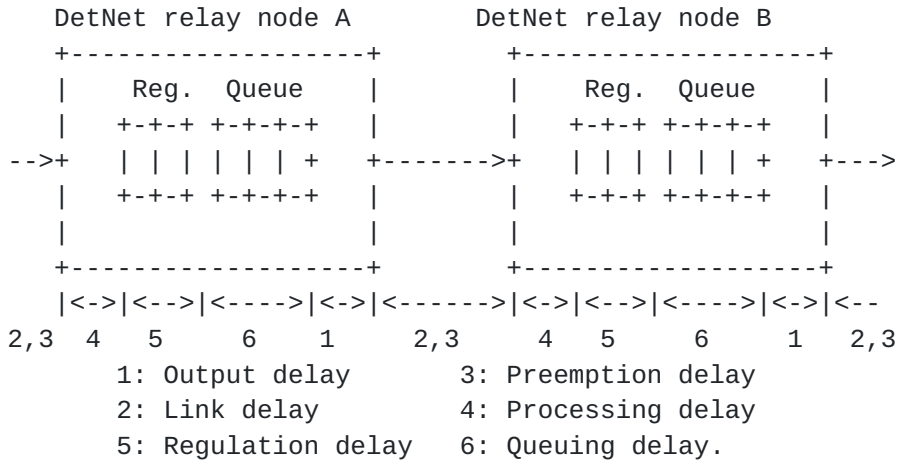

Figure 1: Timing model for DetNet or TSN

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].

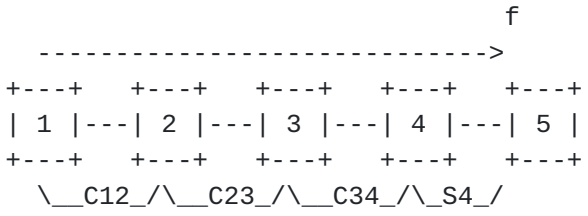

Figure 2: End-to-end latency computation example

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.

- 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.

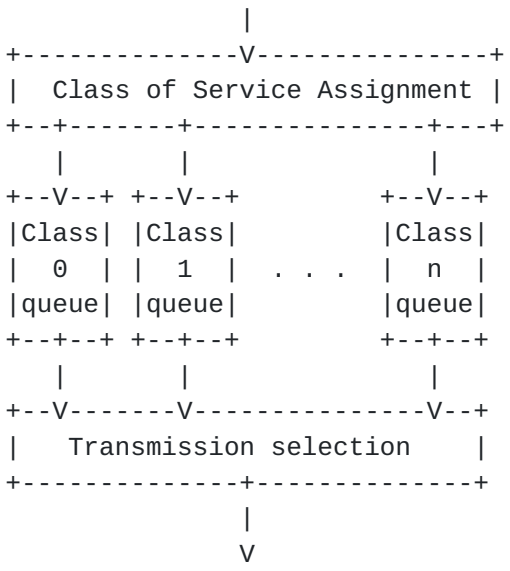

Figure 3: IEEE 802.1Q Queuing Model: Data flow

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.

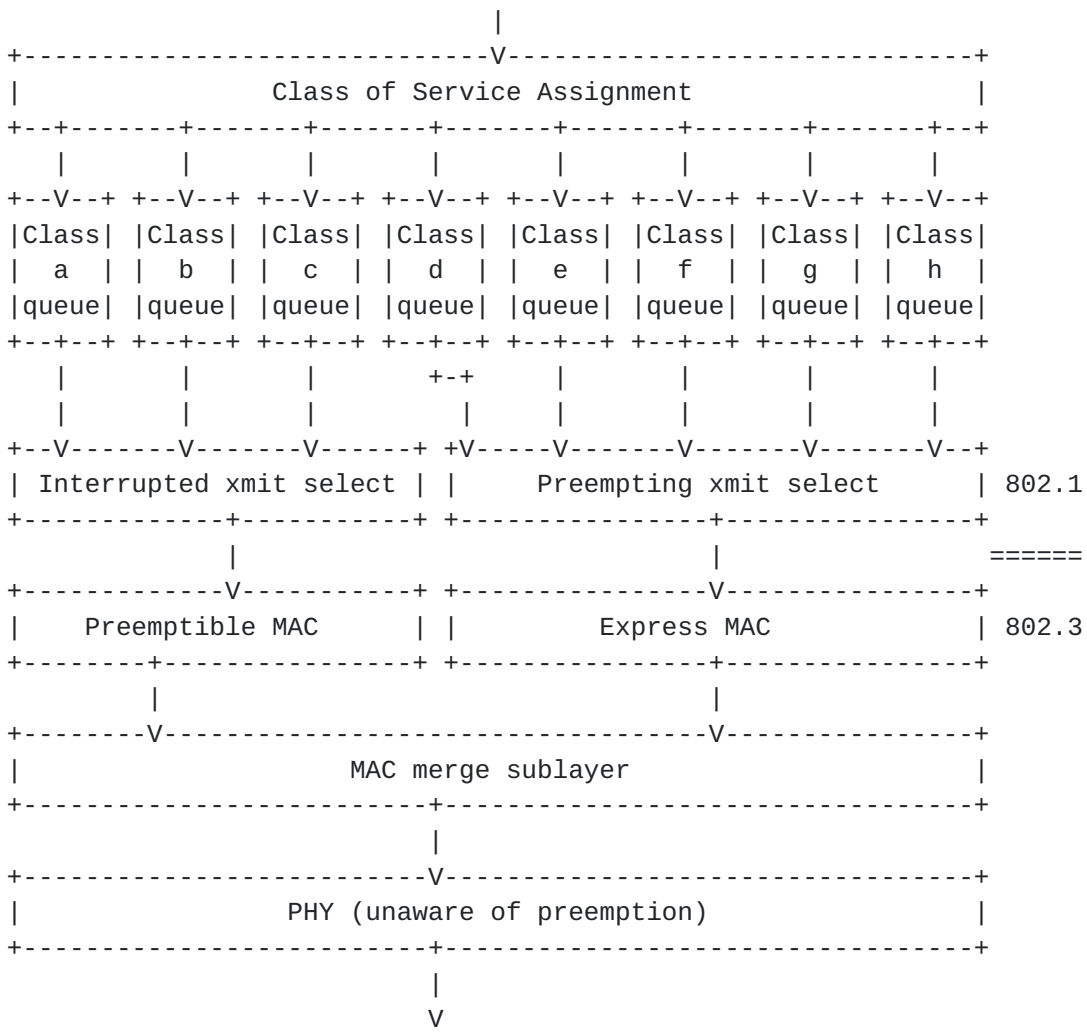

Figure 4: IEEE 802.1Q Queuing Model: Data flow with preemption

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.

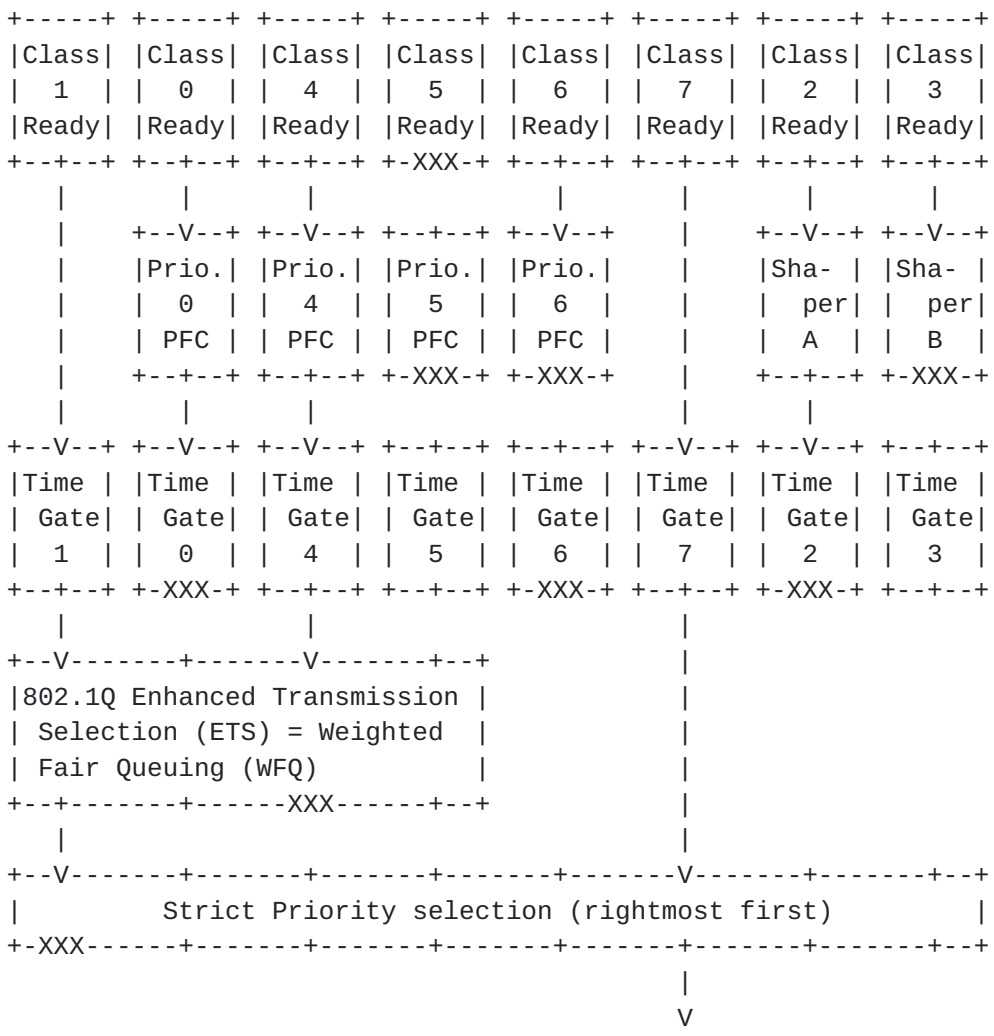

Figure 5: 802.1Q Transmission Selection

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].

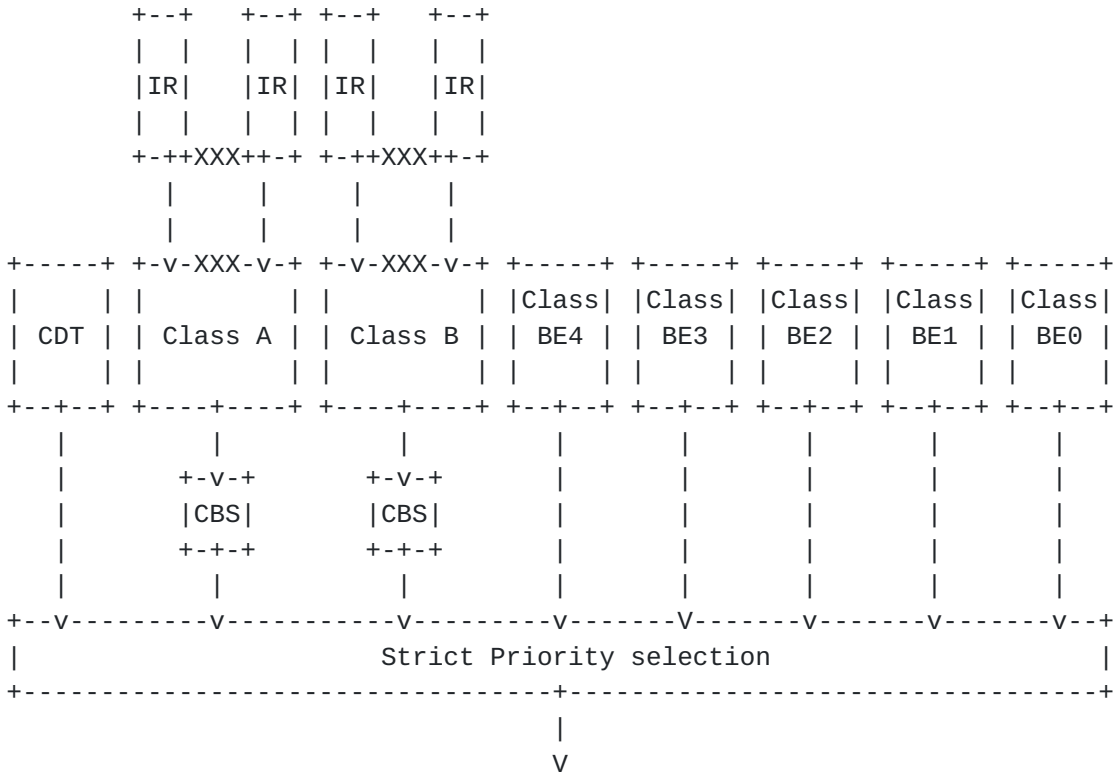

Figure 6: Architecture of one TSN node output port with interleaved regulators (IRs)

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