ï»¿

Internet Engineering Task Force (IETF)
Request for Comments: 9320
Category: Informational
ISSN: 2070-1721

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

November 2022

Deterministic Networking (DetNet) Bounded Latency

Abstract

This document presents a timing model for sources, destinations, and Deterministic Networking (DetNet) transit nodes. Using the model, it provides a methodology to compute end-to-end latency and backlog bounds for various queuing methods. The methodology can be used by the management and control planes and by resource reservation algorithms to provide bounded latency and zero congestion loss for the DetNet service.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9320.

Table of Contents

1. Introduction

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

As explained in [RFC8655], DetNet flows are notably 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. In this document, it is assumed that the paths of DetNet flows are fixed. Before the transmission of a DetNet flow, it is possible to calculate end-to-end latency bounds and the amount of buffer space required at each hop to ensure zero congestion loss; this can be used by the applications identified in [RFC8578].

This document presents a timing model for sources, destinations, and the DetNet transit nodes; using this model, it provides a methodology to compute end-to-end latency and backlog bounds for various queuing mechanisms that can be used by the management and control planes to provide DetNet qualities of service. The methodology used in this document accounts for the possibility of packet reordering within a DetNet node. The bounds on the amount of packet reordering is out of the scope of this document and can be found in [PacketReorderingBounds]. Moreover, this document references specific queuing mechanisms, mentioned in [RFC8655], as proofs of concept that can be used to control packet transmission at each output port and achieve the DetNet quality of service (QoS).

Using the model presented in this document, it is possible for an implementer, user, or standards development organization to select a 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. Section 7 provides an example application of the timing model introduced in this document on a DetNet IP network with a combination of different queuing mechanisms.

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

2. Terminology and Definitions

This document uses the terms defined in [RFC8655]. Moreover, the following terms are used in this document:

T-SPEC
: TrafficSpecification, as defined in Section 5.5 of [RFC9016].

arrival curve
: An arrival curve function alpha(t) is an upper bound on the number of bits seen at an observation point within any time interval t.

CQF
: Cyclic Queuing and Forwarding.

CBS
: Credit-Based Shaper.

TSN
: Time-Sensitive Networking.

PREOF
: A collective name for Packet Replication, Elimination, and Ordering Functions.

POF
: A Packet Ordering Function is a function that reorders packets within a DetNet flow that are received out of order. This function can be implemented by a DetNet edge node, a DetNet relay node, or an end system.

3. DetNet Bounded Latency Model

3.1. Flow Admission

This document assumes that the following paradigm is used to admit DetNet flows:

1. Perform any configuration required by the DetNet transit nodes in the network for aggregates of DetNet flows. This configuration is done beforehand and not tied to any particular DetNet flow.

2. Characterize the new DetNet flow, particularly in terms of required bandwidth.

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. Compute the worst-case end-to-end latency for the DetNet flow using one of the methods below (Sections 3.1.1 and 3.1.2). In the process, determine whether sufficient resources are available for the DetNet flow to guarantee the required latency and to provide zero congestion loss.

5. Assuming that the resources are available, commit those resources to the DetNet flow. This may require adjusting the parameters that control the filtering and/or queuing mechanisms at each hop along the DetNet flow's path.

This paradigm can be implemented using peer-to-peer protocols or using a central controller. In some situations, a lack of resources can require backtracking and recursing through the above list.

Issues, such as service preemption of a DetNet flow in favor of another, when resources are scarce, are not considered here. Also not addressed is the question of how to choose the path to be taken by a DetNet flow.

3.1.1. Static Latency Calculation

The static problem:

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

In this calculation, all of the DetNet flows are known before the calculation commences. This problem is of interest to relatively static networks or static parts of larger networks. It provides bounds on latency and buffer size. The calculations can be extended to provide global optimizations, such as altering the path of one DetNet flow in order to make resources available to another DetNet flow with tighter constraints.

This calculation may be more difficult to perform than the dynamic calculation (Section 3.1.2) because the DetNet flows passing through one port on a DetNet transit node affect each other's latency. The effects can even be circular, from node A to B to C and back to A. On the other hand, the static calculation can often accommodate queuing methods, such as transmission selection by strict priority, that are unsuitable for the dynamic calculation.

3.1.2. Dynamic Latency Calculation

The dynamic problem:

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

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

Dynamic latency calculation can be done based on the static one described in Section 3.1.1; when a new DetNet flow is created or deleted, the entire calculation for all DetNet flows is repeated. If an already-established DetNet flow would be pushed beyond its latency requirements by the new DetNet flow request, then the new DetNet flow request can be refused or some other suitable action can be taken.

The choice of queuing methods is critical to the applicability of the dynamic calculation. Some queuing methods (e.g., CQF, Section 6.6) make it easy to configure bounds on the network's capacity and to make independent calculations for each DetNet flow. Some other queuing methods (e.g., strict priority with the credit-based shaper defined in Section 8.6.8.2 of [IEEE8021Q]) can be used for dynamic DetNet flow creation but yield poorer latency and buffer space guarantees than when that same queuing method is used for static DetNet flow creation (Section 3.1.1).

3.2. Relay Node Model

A model for the operation of a DetNet transit node is required in

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

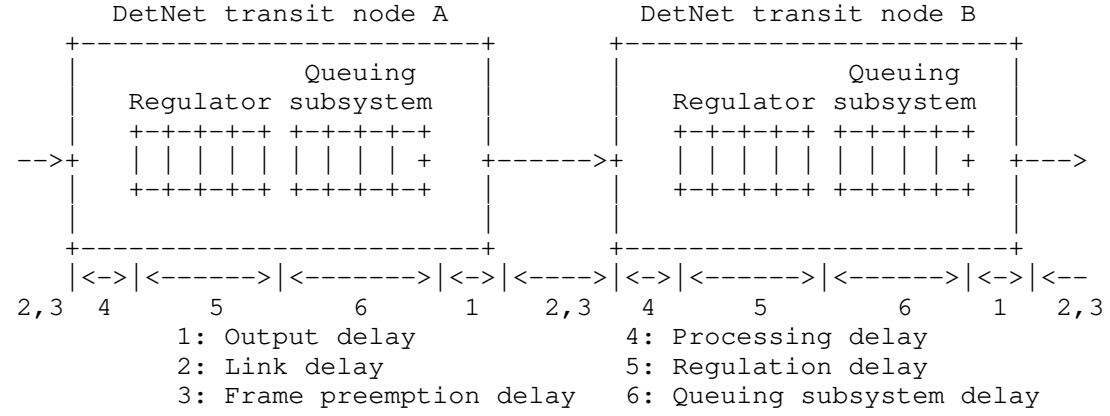

Figure 1: Timing Model for DetNet or TSN

In Figure 1, we see two DetNet transit nodes that are connected via a link. In this model, the only queues that 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

   This is 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. However, in many implementations, a multiplexed connection separates the queuing mechanism from a multi-port Network Interface Card (NIC). This causes variations in the output delay that are hard for the forwarding node to predict or control.

2. Link delay

   This is 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 frame 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 constant. However, a virtual "link" could exhibit a variable link delay.

3. Frame preemption delay

   If the packet is interrupted in order to transmit another packet or packets (e.g., frame preemption, as in [IEEE8023], clause 99), 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 regulator), as shown in Figure 1. This delay can be variable and depends on the details of the operation of the forwarding node.

5. Regulator queuing delay

   A regulator, also known as shaper in [RFC2475], delays some or all of the packets in a traffic stream in order to bring the stream into compliance with an arrival curve; an arrival curve 'alpha(t)' is an upper bound on the number of bits observed

within any interval t. The regulator delay 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.

In this analysis, the measurement is from the point at which a packet is selected for output in a node to the point at which it is selected for output in the next downstream node (i.e., the definition of a "hop"). 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 delays 1-4 are not constant, then additional buffers are required in the queue to absorb these variations. Thus:

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

4. Computing End-to-End Delay Bounds

4.1. Non-queuing Delay Bound

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

```
end_to_end_delay_bound = non_queuing_delay_bound +
queuing_delay_bound
```

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_delay_bound[h] on the sum of the bounds over delays 1, 2, 3, and 4 of Figure 1. These upper bounds are expected to depend on the specific technology of the DetNet transit node at the h-th hop but not on the T-SPEC of this DetNet flow [RFC9016]. Then, set non_queuing_delay_bound = the sum of per-hop_non_queuing_delay_bound[h] over all hops h.

Second, compute queuing_delay_bound as an upper bound to the sum of the queuing delays along the path. The value of queuing_delay_bound depends on the information on the arrival curve of this DetNet flow

and possibly of other flows in the network, as well as the specifics of the queuing mechanisms deployed along the path of this DetNet flow. Note that arrival curve of the DetNet flow at the source is immediately specified by the T-SPEC of this flow. The computation of queuing_delay_bound is described in Section 4.2 as a separate section.

4.2. Queuing Delay Bound

For several queuing mechanisms, queuing_delay_bound is less than the sum of upper bounds on the queuing delays (5 and 6) at every hop. This occurs with (1) per-flow queuing and (2) aggregate queuing with regulators, as explained in Sections 4.2.1, 4.2.2, and 6. For other queuing mechanisms, the only available value of queuing_delay_bound is the sum of the per-hop queuing delay bounds.

The computation of per-hop queuing delay bounds must account for the fact that the arrival curve of a DetNet flow is no longer satisfied at the ingress of a hop, since burstiness increases as one flow traverses one DetNet transit node. If a regulator is placed at a hop, an arrival curve of a DetNet flow at the entrance of the queuing subsystem of this hop is the one configured at the regulator (also called shaping curve in [NetCalBook]); otherwise, an arrival curve of the flow can be derived using the delay jitter of the flow from the last regulation point (the last regulator in the path of the flow if there is any, otherwise the source of the flow) to the ingress of the hop; more formally, assume a DetNet flow has an arrival curve at the last regulation point equal to 'alpha(t)' and the delay jitter from the last regulation point to the ingress of the hop is 'V'. Then, the arrival curve at the ingress of the hop is 'alpha(t+V)'.

For example, consider a DetNet flow with T-SPEC "Interval: tau, MaxPacketsPerInterval: K, MaxPayloadSize: L" at the source. Then, a leaky-bucket arrival curve for such flow at the source is alpha(t)=r * t+ b, t>0; alpha(0)=0, where r is the rate and b is the bucket size, computed as

$$r = K * (L+L') / tau,$$

$$b = K * (L+L').$$

where L' is the size of any added networking technology-specific encapsulation (e.g., MPLS label(s), UDP, or IP headers). Now, if the flow has a delay jitter of 'V' from the last regulation point to the ingress of a hop, an arrival curve at this point is r * t + b + r * V, implying that the burstiness is increased by r*V. More detailed information on arrival curves is available in [NetCalBook].

4.2.1. Per-Flow Queuing Mechanisms

With such mechanisms, each flow uses a separate queue inside every node. The service for each queue is abstracted with a guaranteed rate and a latency. For every DetNet flow, a per-node latency bound, as well as an end-to-end latency bound, can be computed from the traffic specification of this DetNet flow at its source and from the values of rates and latencies at all nodes along its path. An instance of per-flow queuing is Guaranteed Service [RFC2212], for which the details of latency bound calculation are presented in Section 6.5.

4.2.2. Aggregate Queuing Mechanisms

With such mechanisms, multiple flows are aggregated into macro-flows and there is one FIFO queue per macro-flow. A practical example is the credit-based shaper defined in Section 8.6.8.2 of [IEEE8021Q], where a macro-flow is called a "class". One key issue in this context is how to deal with the burstiness cascade; individual flows that share a resource dedicated to a macro-flow may see their burstiness increase, which may in turn cause increased burstiness to other flows downstream of this resource. Computing delay upper bounds for such cases is difficult and, in some conditions,

impossible [CharnyDelay] [BennettDelay]. Also, when bounds are obtained, they depend on the complete configuration and must be recomputed when one flow is added (i.e., the dynamic calculation in Section 3.1.2).

A solution to deal with this issue for the DetNet flows is to reshape them 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 aggregate queuing. An alternative is the interleaved regulator, which reshapes individual DetNet flows without per-flow queuing [SpechtUBS] [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 a per-flow or interleaved regulator is that it does not increase worst-case latency bounds [LeBoudecTheory]. Specifically, when an interleaved regulator is appended to a FIFO subsystem, it does not increase the worst-case delay of the latter. In Figure 1, when the order of packets from the output of a queuing subsystem at node A to the entrance of a regulator at node B is preserved, then the regulator does not increase the worst-case latency bounds. This is made possible if all the systems are FIFO or a DetNet Packet Ordering Function (POF) is implemented just before the regulator. This property does not hold if packet reordering occurs from the output of a queuing subsystem to the entrance of the next downstream interleaved regulator, e.g., at a non-FIFO switching fabric.

Figure 2 shows an example of a network with 5 nodes, an aggregate queuing mechanism, and interleaved regulators, as in Figure 1. An end-to-end delay bound for DetNet 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 delay of the queuing subsystem in node i and interleaved regulator of node j, and S4 is a bound on the delay of the queuing subsystem in node 4 for DetNet flow f. In fact, using the delay definitions in Section 3.2, Cij is a bound on a sum of delays 1, 2, 3, and 6 of node i and delays 4 and 5 of node j. Similarly, S4 is a bound on sum of delays 1, 2, 3, and 6 of node 4. A practical example of the queuing model and delay calculation is presented Section 6.4.

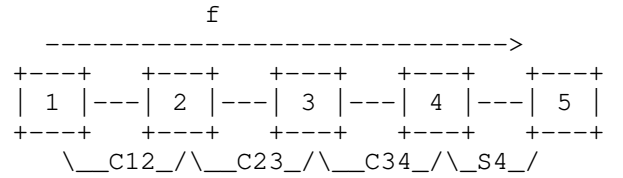

Figure 2: End-to-End Delay Computation Example

If packet reordering does not occur, the end-to-end latency bound calculation provided here gives a tighter latency upper bound than would be obtained by adding the latency bounds of each node in the path of a DetNet flow [TSNwithATS].

4.3. Ingress Considerations

A sender can be a DetNet node that uses exactly the same queuing methods as its adjacent DetNet transit node so that the latency and buffer bounds calculations at the first hop are indistinguishable from those at a later hop within the DetNet domain. On the other hand, the sender may be DetNet unaware; in which case, some conditioning of the DetNet flow may be necessary at the ingress DetNet transit node. The ingress conditioning typically consists of the regulators described in Section 3.2.

4.4. Interspersed DetNet-Unaware Transit Nodes

It is sometimes desirable to build a network that has both DetNet-

aware transit nodes and DetNet-unaware transit nodes and for a DetNet flow to traverse an island of DetNet-unaware transit nodes while still allowing the network to offer delay and congestion loss guarantees. This is possible under certain conditions.

In general, when passing through a DetNet-unaware island, the island may cause delay variation in excess of what would be caused by DetNet nodes. That is, the DetNet flow might be "lumpier" after traversing the DetNet-unaware island. DetNet guarantees for delay and buffer requirements can still be calculated and met if and only if the following are true:

1. The latency variation across the DetNet-unaware island must be bounded and calculable.

2. An ingress conditioning function (Section 4.3) is required at the reentry to the DetNet-aware domain. This will, at least, require some extra buffering to accommodate the additional delay variation and thus further increases the latency bound.

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

5. Achieving Zero Congestion Loss

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

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 flow or an aggregate of flows, we need one buffer space for the packet in transmission, plus space for the packets that are waiting to be selected for output.

Let

* total_in_rate be the sum of the line rates of all input ports that send traffic 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 of input ports that send traffic to this output port.

* max_packet_length be the maximum packet size for packets that may be sent to this output port. This is counted in data units.

* max_delay456 be an upper bound, in seconds, on the sum of the processing delay (4) and the queuing delays (5 and 6) for any packet at this output port.

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

```
backlog_bound = (nb_input_ports * max_packet_length) +
(total_in_rate * max_delay456)
```

The above bound is over the backlog caused by the traffic entering the queue from the input ports of a DetNet node. If the DetNet node also generates packets (e.g., creation of new packets or replication of arriving packets), the bound must accordingly incorporate the introduced backlog.

6. Queuing Techniques

In this section, we present a general queuing data model, as well as some examples of queuing mechanisms. For simplicity of latency bound computation, we assume a leaky-bucket arrival curve for each DetNet flow at the source. Also, at each DetNet transit node, the service for each queue is abstracted with a minimum guaranteed rate and a latency [NetCalBook].

6.1. Queuing Data Model

Sophisticated queuing mechanisms are available in Layer 3 (L3) (e.g., see [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 FIFO. They are not the province solely of bridges; they are an essential part of any DetNet transit node. As illustrated by numerous implementation examples, some of the "Layer 3" mechanisms described in documents, such as [RFC7806], are often integrated in an implementation, with the "Layer 2" mechanisms also implemented in the same node. An integrated model is needed in order to successfully predict the interactions among the different queuing mechanisms needed in a network carrying both DetNet flows and non-DetNet flows.

Figure 3 shows the general model for the flow of packets through the queues of a DetNet transit node. The DetNet packets are mapped to a number of regulators. Here, we assume that the Packet Replication, Elimination, and Ordering Functions (PREOF) are performed before the DetNet packets enter the regulators. All packets are assigned to a set of queues. Packets compete for the selection to be passed to queues in the queuing subsystem. Packets again are selected for output from the queuing subsystem.

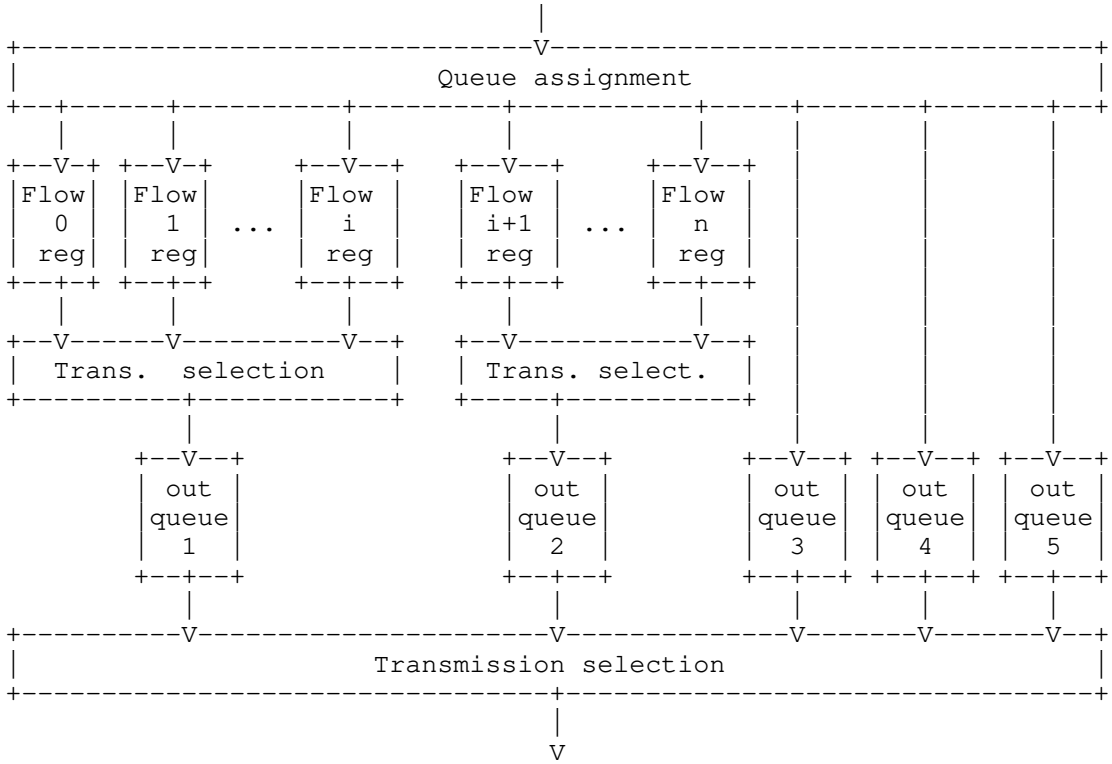

Figure 3: IEEE 802.1Q Queuing Model: Data Flow

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

* discarding packets because a queue is full

* discarding packets marked "yellow" by a metering function in preference to discarding "green" packets [RFC2697]

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

The queue assignment function can be quite complex, even in a bridge [IEEE8021Q], because of the introduction of per-stream filtering and policing ([IEEE8021Q], clause 8.6.5.1). In addition to the Layer 2 priority expressed in the 802.1Q VLAN tag, a DetNet transit node can utilize the information from the non-exhaustive list below to assign a packet to a particular 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, and Differentiated Services Code Point (DSCP) [RFC8939] [RFC8964]

* the queue assignment function can contain metering and policing functions

* MPLS and/or pseudowire labels [RFC6658]

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

6.2. Frame Preemption

In [IEEE8021Q] and [IEEE8023], the transmission of a frame can be interrupted by one or more "express" frames; then, the interrupted frame can continue transmission. The frame preemption is modeled as consisting of two MAC/PHY stacks: one for packets that can be interrupted and one for packets that can interrupt the interruptible packets. Only one layer of frame preemption is supported -- a transmitter cannot have more than one interrupted frame in progress. DetNet flows typically pass through the interrupting MAC. For those DetNet flows with T-SPEC, latency bounds can be calculated by the methods provided in the following sections that account for the effect of frame preemption, according to the specific queuing mechanism that is used in DetNet nodes. Best-effort queues pass through the interruptible MAC and can thus be preempted.

6.3. Time-Aware Shaper

In [IEEE8021Q], the notion of time-scheduling queue gates is described in Section 8.6.8.4. On each node, the transmission selection for packets is controlled by time-synchronized gates; each output queue is associated with a gate. The gates can be either open or closed. The states of the gates are determined by the gate control list (GCL). The GCL specifies the opening and closing times of the gates. The design of the GCL must satisfy the requirement of latency upper bounds of all DetNet flows; therefore, those DetNet flows that traverse a network that uses this kind of shaper must have bounded latency if the traffic and nodes are conformant.

Note that scheduled traffic service relies on a synchronized network and coordinated GCL configuration. Synthesis of the GCL on multiple nodes in a network is a scheduling problem considering all DetNet flows traversing the network, which is a nondeterministic polynomial-time hard (NP-hard) problem [Sch8021Qbv]. Also, at the time of writing, scheduled traffic service supports no more than eight traffic queues, typically using up to seven priority queues and at least one best effort.

6.4. Credit-Based Shaper with Asynchronous Traffic Shaping

In this queuing model, it is assumed that the DetNet nodes are FIFO. We consider the four traffic classes (Definition 3.268 of [IEEE8021Q]): control-data traffic (CDT), class A, class B, and best effort (BE) in decreasing order of priority. Flows of classes A and B are DetNet flows that are less critical than CDT (such as studio audio and video traffic, as in IEEE 802.1BA Audio-Video-Bridging). This model is a subset of Time-Sensitive Networking, as described next.

Based on the timing model described in Figure 1, contention occurs only at the output port of a DetNet transit node; therefore, the focus of the rest of this subsection is on the regulator and queuing subsystem in the output port of a DetNet transit node. The input flows are identified using the information in (Section 5.1 of [RFC8939]). Then, they are aggregated into eight macro-flows based on their service requirements; we refer to each macro-flow as a class. The output port performs aggregate scheduling with eight queues (queuing subsystems): one for CDT, one for class A flows, one for class B flows, and five for BE traffic denoted as BE0-BE4. The queuing policy for each queuing subsystem is FIFO. In addition, each node output port also performs per-flow regulation for class A and B flows using an interleaved regulator (IR). This regulation is called asynchronous traffic shaping [IEEE8021Qcr]. Thus, at each output port of a node, there is one interleaved regulator per input port and per class; the interleaved regulator is mapped to the regulator depicted in Figure 1. 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 queuing subsystem; there is no regulator for CDT and BE flows.

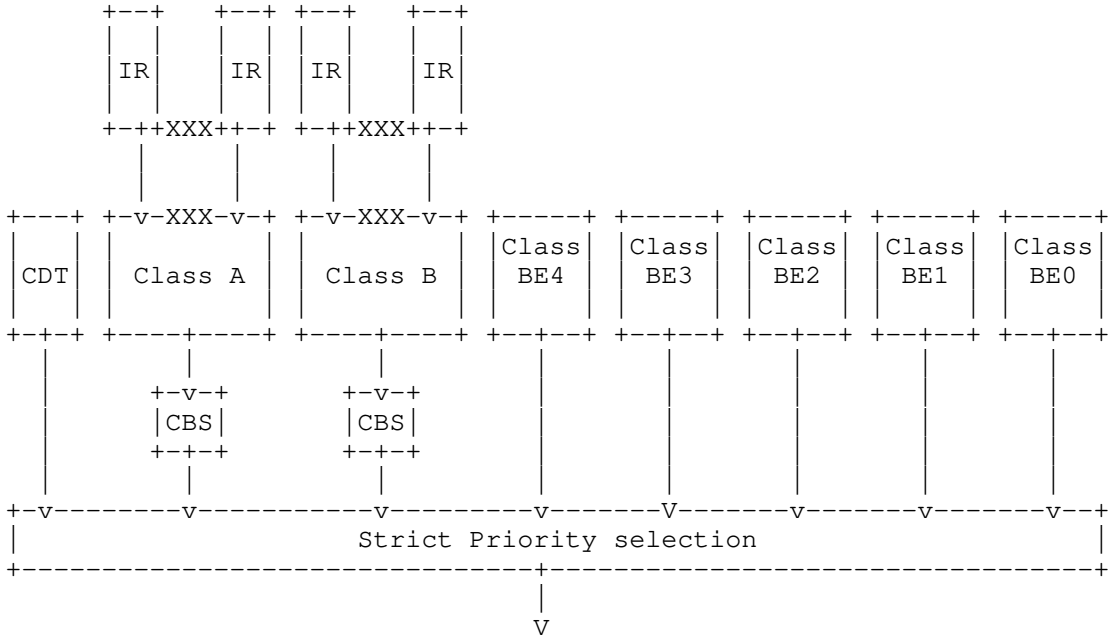

Figure 4: The Architecture of an Output Port inside a Relay Node with Interleaved Regulators (IRs) and a Credit-Based Shaper (CBS)

Each of the queuing subsystems for classes A and B contains a credit-based shaper (CBS). The CBS serves a packet from a class according to the available credit for that class. As described in Section 8.6.8.2 and Annex L.1 of [IEEE8021Q], the credit for each class A or B increases based on the idle slope (as guaranteed rate) and decreases based on the sendslope (typically equal to the difference between the guaranteed and the output link rates), both of which are parameters of the CBS. The CDT and BE0-BE4 flows are served by separate queuing 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 class A and B traffic can be provided only if CDT is bounded. It is assumed that the CDT has a leaky-bucket arrival curve with two parameters: r_h as rate and b_h as bucket size. That is, the amount of bits entering a node within a time interval t is bounded by r_h * t + b_h.

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

At each DetNet transit node implementing an interleaved regulator, packets of multiple flows are processed in one FIFO queue. The packet at the head of the queue is regulated based on its leaky-bucket parameters. It is released at the earliest time at which this is possible without violating the constraint.

The regulation parameters for a flow (leaky-bucket rate and bucket size) are the same at its source and at all DetNet transit nodes along its path in the case where all clocks are perfect. However, in reality, there is clock non-ideality throughout the DetNet domain, even with clock synchronization. This phenomenon causes inaccuracy in the rates configured at the regulators that may lead to network instability. To avoid instability, the rates are set as the source rates with some positive margin when configuring regulators. [ThomasTime] describes and provides solutions to this issue.

6.4.1. Delay Bound Calculation

A delay bound of the queuing subsystem ((4) in Figure 1) of a given DetNet node for a flow of class A or B can be computed if the following condition holds:

> The sum of leaky-bucket rates of all flows of this class at this transit node <= R, where R is given below for every class

If the condition holds, the delay bounds for a flow of class X (A or B) is d_X and calculated as:

$$d_X = T_X + (b_t_X - L_min_X)/R_X - L_min_X/c$$

where L_min_X is the minimum packet lengths of class X (A or B); c is the output link transmission rate; and b_t_X is the sum of the b term (bucket size) for all the flows of the class X. Parameters R_X and T_X are calculated as follows for class A and B, separately.

If the flow is of class A:

$$R_A = I_A * (c - r_h)/ c$$

$$T_A = (L_nA + b_h + r_h * L_n/c)/(c - r_h)$$

where I_A is the idle slope for class A; L_nA is the maximum packet length of class B and BE packets; L_n is the maximum packet length of classes A, B, and BE; and r_h is the rate and b_h is the bucket size of CDT leaky-bucket arrival curve.

If the flow is of class B:

$$R_B = I_B * (c - r_h)/ c$$

$$T_B = (L_BE + L_A + L_nA * I_A/(c_h - I_A) + b_h + r_h * L_n/ c)/(c - r_h)$$

where I_B is the idle slope for class B; L_A is the maximum packet length of class A; and L_BE is the maximum packet length of class BE.

Then, as discussed in Section 4.2.2, an interleaved regulator does not increase the delay bound of the upstream queuing subsystem;

therefore, an end-to-end delay bound for a DetNet flow of class X (A or B) is the sum of d_X_i for all node i in the path of the flow, where d_X_i is the delay bound of queuing subsystem in node i, which is computed as above. According to the notation in Section 4.2.2, the delay bound of the queuing subsystem in a node i and interleaved regulator in node j, i.e., Cij, is:

$$C_{ij} = d_{X\_i}$$

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

6.4.2. Flow Admission

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

The static admission control:

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

The dynamic admission control:

> For dynamic admission control, we allocate a static value for rate (R) and a maximum bucket size (b_t) to every node and each class A or B. In addition, for every node and each class A or B, two counters are maintained:
>
> R_acc is equal to the sum of the leaky-bucket rates of all flows of this class already admitted at this node; at all times, we must have:
>
> $$R\_acc \leq R, \quad \text{(Eq. 1)}$$
>
> b_acc is equal to the sum of the bucket sizes of all flows of this class already admitted at this node; at all times, we must have:
>
> $$b\_acc \leq b\_t. \quad \text{(Eq. 2)}$$
>
> A new class A or B flow is admitted at this node if Eqs. (1) and (2) continue to be satisfied after adding its leaky-bucket rate and bucket size to R_acc and b_acc. A class A or B flow is admitted in the network if it is admitted at all nodes along its path. When this happens, all variables R_acc and b_acc along its path must be incremented to reflect the addition of the flow. Similarly, when a class A or B flow leaves the network, all variables R_acc and b_acc along its path must be decremented to reflect the removal of the flow.

The choice of the static values of R and b_t at all nodes and classes must be done in a prior configuration phase: R controls the bandwidth allocated to this class at this node, and b_t affects the delay bound and the buffer requirement. The value of R must be set such that

$$R \leq I_X*(c-r_h)/c$$

where I_X is the idleslope of credit-based shaper for class X={A,B},

c is the transmission rate of the output link, and r_h is the leaky-bucket rate of the CDT class.

6.5. Guaranteed Service

The Guaranteed Service is defined in [RFC2212]. The flow, at the source, has a leaky-bucket arrival curve with two parameters: r as rate and b as bucket size, i.e., the amount of bits entering a node within a time interval t is bounded by r * t + b.

If a resource reservation on a path is applied, a node provides a guaranteed rate R and maximum service latency of T. This can be interpreted in a way that the bits might have to wait up to T before being served with a rate greater or equal to R. The delay bound of the flow traversing the node is T + b / R.

Consider a Guaranteed Service [RFC2212] path including a sequence of nodes, where the i-th node provides a guaranteed rate R_i and maximum service latency of T_i. Then, the end-to-end delay bound for a flow on this can be calculated as sum(T_i) + b / min(R_i).

The provided delay bound is based on a simple case of Guaranteed Service, where only a guaranteed rate and maximum service latency and a leaky-bucket arrival curve are available. If more information about the flow is known, e.g., the peak rate, the delay bound is more complicated; the details are available in [RFC2212] and Section 1.4.1 of [NetCalBook].

6.6. Cyclic Queuing and Forwarding

Annex T of [IEEE8021Q] describes Cyclic Queuing and Forwarding (CQF), which provides bounded latency and zero congestion loss using the time-scheduled gates of Section 8.6.8.4 of [IEEE8021Q]. For a given class of DetNet flows, a set of two or more buffers is provided at the output queue layer of Figure 3. A cycle time T_c is configured for each class of DetNet flows c, and all of the buffer sets in a class of DetNet flows swap buffers simultaneously throughout the DetNet domain at that cycle rate, all in phase. In such a mechanism, the regulator, as mentioned in Figure 1, is not required.

In the case of two-buffer CQF, each class of DetNet flows c has two buffers, namely buffer1 and buffer2. In a cycle (i) when buffer1 accumulates received packets from the node's reception ports, buffer2 transmits the already stored packets from the previous cycle (i-1). In the next cycle (i+1), buffer2 stores the received packets and buffer1 transmits the packets received in cycle (i). The duration of each cycle is T_c.

The cycle time T_c must be carefully chosen; it needs to be large enough to accommodate all the DetNet traffic, plus at least one maximum packet (or fragment) size from lower priority queues, which might be received within a cycle. Also, the value of T_c includes a time interval, called dead time (DT), which is the sum of delays 1, 2, 3, and 4 defined in Figure 1. The value of DT guarantees that the last packet of one cycle in a node is fully delivered to a buffer of the next node in the same cycle. A two-buffer CQF is recommended if DT is small compared to T_c. For a large DT, CQF with more buffers can be used, and a cycle identification label can be added to the packets.

The per-hop latency is determined by the cycle time T_c: a packet transmitted from a node at a cycle (i) is transmitted from the next node at cycle (i+1). Then, if the packet traverses h hops, the maximum latency experienced by the packet is from the beginning of cycle (i) to the end of cycle (i+h); also, the minimum latency is from the end of cycle (i), before the DT, to the beginning of cycle (i+h). Then, the maximum latency is:

$$(h+1)\ T\_c$$

and the minimum latency is:

(h-1) T_c + DT.

Ingress conditioning (Section 4.3) may be required if the source of a DetNet flow does not itself employ CQF. Since there are no per-flow parameters in the CQF technique, per-hop configuration is not required in the CQF forwarding nodes.

7. Example Application on DetNet IP Network

This section provides an example application of the timing model presented in this document to control the admission of a DetNet flow on a DetNet-enabled IP network. Consider Figure 5, taken from Section 3 of [RFC8939], which shows a simple IP network:

* End system 1 implements Guaranteed Service [RFC2212], as in Section 6.5, between itself and relay node 1.

* Sub-network 1 is a TSN network. The nodes in sub-network 1 implement credit-based shapers with asynchronous traffic shaping, as in Section 6.4.

* Sub-network 2 is a TSN network. The nodes in sub-network 2 implement Cyclic Queuing and Forwarding with two buffers, as in Section 6.6.

* The relay nodes 1 and 2 implement credit-based shapers with asynchronous traffic shaping, as in Section 6.4. They also perform the aggregation and mapping of IP DetNet flows to TSN streams (Section 4.4 of [RFC9023]).

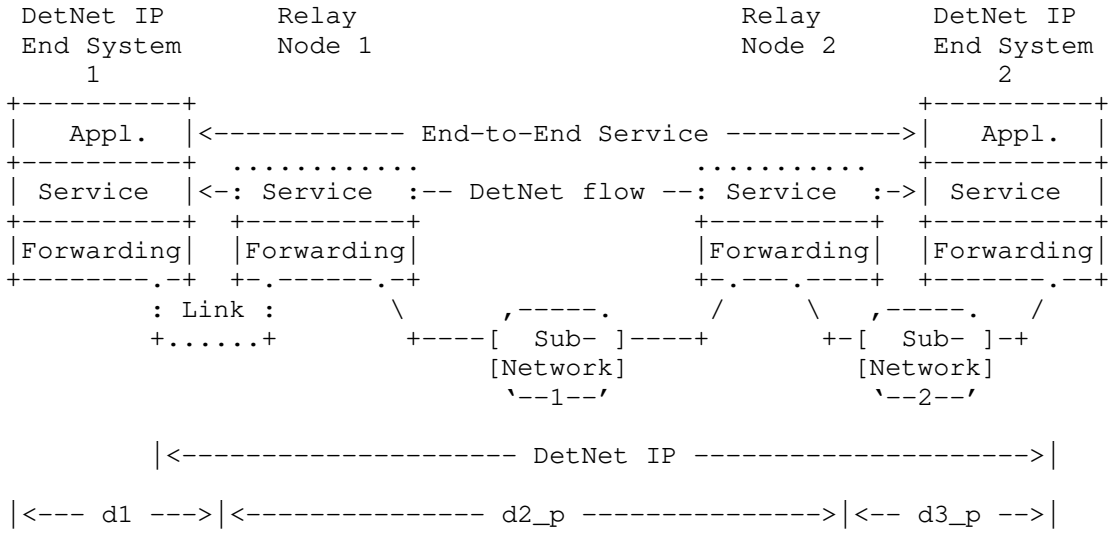

Figure 5: A Simple DetNet-Enabled IP Network, Taken from RFC 8939

Consider a fully centralized control plane for the network of Figure 5, as described in Section 3.2 of [DETNET-CONTROL-PLANE]. Suppose end system 1 wants to create a DetNet flow with a traffic specification destined to end system 2 with end-to-end delay bound requirement D. Therefore, the control plane receives a flow establishment request and calculates a number of valid paths through the network (Section 3.2 of [DETNET-CONTROL-PLANE]). To select a proper path, the control plane needs to compute an end-to-end delay bound at every node of each selected path p.

The end-to-end delay bound is d1 + d2_p + d3_p, where d1 is the delay bound from end system 1 to the entrance of relay node 1, d2_p is the delay bound for path p from relay node 1 to the entrance of the first node in sub-network 2, and d3_p is the delay bound of path p from the first node in sub-network 2 to end system 2. The computation of d1 is explained in Section 6.5. Since the relay node 1, sub-network 1, and relay node 2 implement aggregate queuing, we use the results in Sections 4.2.2 and 6.4 to compute d2_p for the path p. Finally, d3_p is computed using the delay bound computation of Section 6.6. Any

path p, such that d1 + d2_p + d3_p <= D, satisfies the delay bound requirement of the flow. If there is no such path, the control plane may compute a new set of valid paths and redo the delay bound computation or reject the DetNet flow.

As soon as the control plane selects a path that satisfies the delay bound constraint, it allocates and reserves the resources in the path for the DetNet flow (Section 4.2 of [DETNET-CONTROL-PLANE]).

8. Security Considerations

Detailed security considerations for DetNet are cataloged in [RFC9055], and more general security considerations are described in [RFC8655].

Security aspects that are unique to DetNet are those whose aim is to provide the specific QoS aspects of DetNet, specifically bounded end-to-end delivery latency and zero congestion loss. Achieving such loss rates and bounded latency may not be possible in the face of a highly capable adversary, such as the one envisioned by the Internet Threat Model of BCP 72 [RFC3552], which can arbitrarily drop or delay any or all traffic. In order to present meaningful security considerations, we consider a somewhat weaker attacker who does not control the physical links of the DetNet domain but may have the ability to control or change the behavior of some resources within the boundary of the DetNet domain.

Latency bound calculations use parameters that reflect physical quantities. If an attacker finds a way to change the physical quantities, unknown to the control and management planes, the latency calculations fail and may result in latency violation and/or congestion losses. An example of such attacks is to make some traffic sources under the control of the attacker send more traffic than their assumed T-SPECs. This type of attack is typically avoided by ingress conditioning at the edge of a DetNet domain. However, it must be insured that such ingress conditioning is done per flow and that the buffers are segregated such that if one flow exceeds its T-SPEC, it does not cause buffer overflow for other flows.

Some queuing mechanisms require time synchronization and operate correctly only if the time synchronization works correctly. In the case of CQF, the correct alignments of cycles can fail if an attack against time synchronization fools a node into having an incorrect offset. Some of these attacks can be prevented by cryptographic authentication as in Annex K of [IEEE1588] for the Precision Time Protocol (PTP). However, the attacks that change the physical latency of the links used by the time synchronization protocol are still possible even if the time synchronization protocol is protected by authentication and cryptography [DelayAttack]. Such attacks can be detected only by their effects on latency bound violations and congestion losses, which do not occur in normal DetNet operation.

9. IANA considerations

This document has no IANA actions.

10. References

10.1. Normative References

[IEEE8021Q] IEEE, "IEEE Standard for Local and Metropolitan Area Networks--Bridges and Bridged Networks", IEEE Std 802.1Q-2018, DOI 10.1109/IEEESTD.2018.8403927, July 2018, <https://ieeexplore.ieee.org/document/8403927>.

[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>.

[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>.

[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>.

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/info/rfc8655>.

[RFC8939] Varga, B., Ed., Farkas, J., Berger, L., Fedyk, D., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP", RFC 8939, DOI 10.17487/RFC8939, November 2020, <https://www.rfc-editor.org/info/rfc8939>.

[RFC8964] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., Bryant, S., and J. Korhonen, "Deterministic Networking (DetNet) Data Plane: MPLS", RFC 8964, DOI 10.17487/RFC8964, January 2021, <https://www.rfc-editor.org/info/rfc8964>.

[RFC9016] Varga, B., Farkas, J., Cummings, R., Jiang, Y., and D. Fedyk, "Flow and Service Information Model for Deterministic Networking (DetNet)", RFC 9016, DOI 10.17487/RFC9016, March 2021, <https://www.rfc-editor.org/info/rfc9016>.

10.2. Informative References

[BennettDelay]
Bennett, J. C. R., Benson, K., Charny, A., Courtney, W. F., and J.-Y. Le Boudec, "Delay jitter bounds and packet scale rate guarantee for expedited forwarding", DOI 10.1109/TNET.2002.801404, August 2002, <https://dl.acm.org/citation.cfm?id=581870>.

[CharnyDelay]
Charny, A. and J.-Y. Le Boudec, "Delay Bounds in a Network with Aggregate Scheduling", DOI 10.1007/3-540-39939-9_1, September 2002, <https://link.springer.com/chapter/10.1007/3-540-39939-9_1>.

[DelayAttack]
Barreto, S., Suresh, A., and J. L. Boudec, "Cyber-attack on packet-based time synchronization protocols: The undetectable Delay Box", DOI 10.1109/I2MTC.2016.7520408, May 2016, <https://ieeexplore.ieee.org/document/7520408>.

[DETNET-CONTROL-PLANE]
Malis, A., Geng, A., Ed., Chen, M., Qin, F., and B. Varga, "Deterministic Networking (DetNet) Controller Plane Framework", Work in Progress, Internet-Draft, draft-ietf-detnet-controller-plane-framework-02, 28 June 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-controller-plane-framework-02>.

[IEEE1588] IEEE, "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", IEEE Std 1588-2008, DOI 10.1109/IEEESTD.2008.4579760, July 2008, <https://ieeexplore.ieee.org/document/4579760>.

[IEEE8021Qcr]
IEEE 802.1, "802.1Qcr-2020 - IEEE Standard for Local and

Metropolitan Area Networks--Bridges and Bridged Networks Amendment 34:Asynchronous Traffic Shaping", November 2020, <https://ieeexplore.ieee.org/document/9253013>.

[IEEE8021TSN]
IEEE 802.1, "802.1 Time-Sensitive Networking (TSN) Task Group", <https://1.ieee802.org/tsn/>.

[IEEE8023] IEEE, "IEEE Standard for Ethernet", IEEE Std 802.3-2018, DOI 10.1109/IEEESTD.2018.8457469, August 2018, <http://ieeexplore.ieee.org/document/8457469>.

[LeBoudecTheory]
Le Boudec, J.-Y., "A Theory of Traffic Regulators for Deterministic Networks With Application to Interleaved Regulators", DOI 10.1109/TNET.2018.2875191, November 2018, <https://ieeexplore.ieee.org/document/8519761>.

[NetCalBook]
Le Boudec, J.-Y. and P. Thiran, "Network Calculus: A Theory of Deterministic Queuing Systems for the Internet", Springer Science & Business Media, vol. 2050, 2001, <https://leboudec.github.io/netcal/latex/netCalBook.pdf>.

[PacketReorderingBounds]
Mohammadpour, E. and J.-Y. Le Boudec, "On Packet Reordering in Time-Sensitive Networks", DOI 10.1109/TNET.2021.3129590, December 2021, <https://ieeexplore.ieee.org/document/9640523>.

[RFC2697] Heinanen, J. and R. Guerin, "A Single Rate Three Color Marker", RFC 2697, DOI 10.17487/RFC2697, September 1999, <https://www.rfc-editor.org/info/rfc2697>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC8578] Grossman, E., Ed., "Deterministic Networking Use Cases", RFC 8578, DOI 10.17487/RFC8578, May 2019, <https://www.rfc-editor.org/info/rfc8578>.

[RFC9023] Varga, B., Ed., Farkas, J., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane: IP over IEEE 802.1 Time-Sensitive Networking (TSN)", RFC 9023, DOI 10.17487/RFC9023, June 2021, <https://www.rfc-editor.org/info/rfc9023>.

[RFC9055] Grossman, E., Ed., Mizrahi, T., and A. Hacker, "Deterministic Networking (DetNet) Security Considerations", RFC 9055, DOI 10.17487/RFC9055, June 2021, <https://www.rfc-editor.org/info/rfc9055>.

[Sch8021Qbv]
Craciunas, S., Oliver, R., Chmelik, M., and W. Steiner, "Scheduling Real-Time Communication in IEEE 802.1Qbv Time Sensitive Networks", DOI 10.1145/2997465.2997470, October 2016, <https://dl.acm.org/doi/10.1145/2997465.2997470>.

[SpechtUBS]
Specht, J. and S. Samii, "Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks", DOI 10.1109/ECRTS.2016.27, July 2016, <https://ieeexplore.ieee.org/abstract/document/7557870>.

[ThomasTime]
Thomas, L. and J.-Y. Le Boudec, "On Time Synchronization Issues in Time-Sensitive Networks with Regulators and Nonideal Clocks", DOI 10.1145/3393691.339420, June 2020, <https://dl.acm.org/doi/10.1145/3393691.3394206>.

[TSNwithATS]
Mohammadpour, E., Stai, E., Mohiuddin, M., and J.-Y. Le Boudec, "Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping", DOI 10.1109/ITC30.2018.10053, September 2018, <https://ieeexplore.ieee.org/document/8493026>.

Acknowledgments

We would like to thank Lou Berger, Tony Przygienda, John Scudder, Watson Ladd, Yoshifumi Nishida, Ralf Weber, Robert Sparks, Gyan Mishra, Martin Duke, Ã\211ric Vyncke, Lars Eggert, Roman Danyliw, and Paul Wouters for their useful feedback on this document.

Contributors

RFC 7322 limits the number of authors listed on the front page to a maximum of 5. The editor wishes to thank and acknowledge the following author for contributing text to this document:

Janos Farkas
Ericsson
Email: janos.farkas@ericsson.com

Authors' Addresses

Norman Finn
Huawei Technologies Co. Ltd
3101 Rio Way
Spring Valley, California 91977
United States of America
Phone: +1 925 980 6430
Email: nfinn@nfinnconsulting.com

Jean-Yves Le Boudec
EPFL
IC Station 14
CH-1015 Lausanne
Switzerland
Email: jean-yves.leboudec@epfl.ch

Ehsan Mohammadpour
EPFL
IC Station 14
CH-1015 Lausanne
Switzerland
Email: ehsan.mohammadpour@epfl.ch

Jiayi Zhang
Huawei Technologies Co. Ltd
Q27, No.156 Beiqing Road
Beijing
100095
China
Email: zhangjiayi11@huawei.com

BalÃ¡zs Varga
Ericsson
Budapest
Konyves KÃ¡lmÃ¡n krt. 11/B
1097
Hungary
Email: balazs.a.varga@ericsson.com