< draft-ietf-detnet-bounded-latency-07.txt	draft-ietf-detnet-bounded-latency-07-jgs-markup.txt >
skipping to change at *page 4, line 21*	skipping to change at *page 4, line 21*
met, will enable them to work together.	met, will enable them to work together.
2. Terminology and Definitions	2. Terminology and Definitions
This document uses the terms defined in [RFC8655].	This document uses the terms defined in [RFC8655].
3. DetNet bounded latency model	3. DetNet bounded latency model
3.1. Flow admission	3.1. Flow admission
This document assumes that following paradigm is used to admit DetNet flows:	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.	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.	2. Characterize the new DetNet flow, particularly in terms of required bandwidth.
3. Establish the path that the DetNet flow will take through the	3. Establish the path that the DetNet flow will take through the
skipping to change at *page 5, line 5*	skipping to change at *page 4, line 50*
5. Assuming that the resources are available, commit those resources to the DetNet flow. This may or may not require adjusting the parameters that control the filtering and/or queuing mechanisms at each hop along the DetNet flow's path.	5. Assuming that the resources are available, commit those resources to the DetNet flow. This may or may not 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 this list.	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 this list.
	What is "this list" in the sentence above?
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.	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	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.	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.
skipping to change at *page 5, line 28*	skipping to change at *page 5, line 27*
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 delay 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.	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 delay 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.
The static latency calculation is not limited only to static networks; the entire calculation for all DetNet flows can be repeated	The static latency calculation is not limited only to static networks; the entire calculation for all DetNet flows can be repeated
	This is a surprising sentence and I wonder if it can be expressed some way that doesn't make the reader do the mental equivalent of slipping on a banana peel. I mean, here we have a subsection called "static latency calculation" and a preceding paragraph that says "this problem is of interest to relatively static networks", ok fine... and then in the very next paragraph you say "not limited only to static networks". The reader cries out, please make up your mind!
each time a new DetNet flow is created or deleted. If some already- established DetNet flow would be pushed beyond its latency requirements by the new DetNet flow, then the new DetNet flow can be refused, or some other suitable action taken.	each time a new DetNet flow is created or deleted. If some already- established DetNet flow would be pushed beyond its latency requirements by the new DetNet flow, then the new DetNet flow can be refused, or some other suitable action taken.
This calculation may be more difficult to perform than that of the dynamic calculation (Section 3.1.2), because the DetNet flows passing through one port on a DetNet transit node affect each others' latency. The effects can even be circular, from a 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.	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 a 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	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	The dynamic problem: Given a network whose maximum capacity for DetNet flows is bounded by a set of static configuration parameters applied
skipping to change at *page 7, line 33*	skipping to change at *page 7, line 33*

```
      "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.  [IEEE8023] clause 99 frame preemption) an
      arbitrary delay can result.

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

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

```
      "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.  [IEEE8023] clause 99 frame preemption) an
      arbitrary delay can result.

   4.  Processing delay
      This delay covers the time from the reception of the last bit of
      the packet to the time the packet is enqueued in the regulator
```

```
Is "regulator" a term defined in one of the normative references, or a
term of art so common that the reader can be assumed to be familiar with
it?
```

```
      (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.
```

skipping to change at *page 8, line 13* | skipping to change at *page 8, line 13*

```
      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
```

```
      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
```

```
The first sentence above is challenging for me to make sense of.  My
problem is with "the initial and final" (which implies, two) "measurement
point" (which implies, one).  That is, there's a disagreement in number
in how the sentence is written.  A rewrite may be in order?
```

```
   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
```

skipping to change at *page 9, line 31* | skipping to change at *page 9, line 31*

```
4.2.  Queuing delay bound

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

   For other queuing mechanisms the only available value of
   queuing_delay_bound 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 transit node.

4.2.1.  Per-flow queuing mechanisms

   With such mechanisms, each flow uses a separate queue inside every
   node.  The service for each queue is abstracted with a guaranteed
   rate and a latency.  For every DetNet flow, a per-node delay bound as
   well as an end-to-end delay 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.  The per-flow
   queuing is used in Guaranteed-Service IntServ.  Details of
   calculation for Guaranteed-Service IntServ are described in
```

```
4.2.  Queuing delay bound

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

   For other queuing mechanisms the only available value of
   queuing_delay_bound is the sum of the per-hop queuing delay bounds.
```

```
So far so good...
```

```
   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 transit node.
```

```
This appears to me as though it's a disconnected thought from the
previous sentence. Does it somehow logically follow? Also, it's the case
that you're not telling the reader how to compute those bounds, right?
Just saying "my goodness it's a PITA to compute them" and providing a
tiny hint as to why?
```

```
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 delay bound as
   well as an end-to-end delay 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.  The per-flow
   queuing is used in Guaranteed-Service IntServ.  Details of
   calculation for Guaranteed-Service IntServ are described in
```

skipping to change at *page 11, line 31* | skipping to change at *page 11, line 31*

A sender can be a DetNet node which uses exactly the same queuing methods as its adjacent DetNet transit node, so that the delay 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.

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

4.4. Interspersed DetNet-unaware transit nodes

It is sometimes desirable to build a network that has both DetNet-aware transit nodes and DetNet-uaware 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.

A sender can be a DetNet node which uses exactly the same queuing methods as its adjacent DetNet transit node, so that the delay 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.

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

4.4. Interspersed DetNet-unaware transit nodes

It is sometimes desirable to build a network that has both DetNet-aware transit nodes and DetNet-uaware 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.

skipping to change at *page 13, line 42* skipping to change at *page 13, line 42*

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 PREOF (Packet Replication, Elimination and Ordering Functions) functions 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.

```
                                  |
+--------------------------------V----------------------------------+
|                             Queue assignment                      |
+--+------+----------+---------+-----------+-----+-------+-------+--+
   |      |          |         |           |     |       |       |
```

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 PREOF (Packet Replication, Elimination and Ordering Functions) functions are

This is just a nit, but "... Functions) functions" above scans kind of funny. It's an "ATM machine" problem -- PREOF contains the word "functions" within itself but normally we just use it as a noun "pre-off". When used that way, sure, "PREOF functions" doesn't seem wrong. But when expanding it out, as you've done here (thank you for expanding it by the way) then it's a little funny.

Change it or not, as you prefer.

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.

```
                                  |
+--------------------------------V----------------------------------+
|                             Queue assignment                      |
+--+------+----------+---------+-----------+-----+-------+-------+--+
   |      |          |         |           |     |       |       |
```

skipping to change at *page 15, line 34* skipping to change at *page 15, line 34*

In [IEEE8021Q] and [IEEE8023], the transmission of a frame can be interrupted by one or more "express" frames, and then the interrupted frame can continue transmission. 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 bound can be calculated by the methods provided in the following sections that accounts for the affect 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 close. 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 GCL should satisfy the requirement of latency upper bounds of all DetNet flows; therefore, those DetNet flows traverse a network should have bounded latency, if the traffic and nodes are conformant.

In [IEEE8021Q] and [IEEE8023], the transmission of a frame can be interrupted by one or more "express" frames, and then the interrupted frame can continue transmission. 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 bound can be calculated by the methods provided in the following sections that accounts for the

"methods ... accounts" seems like a disagreement in number. Probably should be "methods ... account", as in "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 GCL should satisfy the requirement of latency upper bounds of all DetNet flows; therefore, those DetNet flows that traverse a network should have bounded latency, if the traffic and nodes are conformant.

Would it be right to insert "that uses this kind of shaper" or similar? As in, "therefore, those DetNet flows that traverse a network that uses this kind of shaper should have a bounded latency".

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 DetNet flows traversing the network, which is a non-deterministic polynomial-time hard (NP-hard) problem [Sch8021Qbv]. Also, at this 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 the considered queuing model, we considered 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 together referred as AVB flows. This model is a subset of Time-Sensitive Networking as described next.

Based on the timing model described in Figure 1, the 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

skipping to change at *page 17, line 5*

Shaper [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 queuing subsytem; there is no regulator for such classes.

```
       +--+   +--+ +--+   +--+
       |  |   |  | |  |   |  |
       |IR|   |IR| |IR|   |IR|
       |  |   |  | |  |   |  |
       +-++XXX++-+ +-++XXX++-+
         |     |     |     |
         |     |     |     |
 +---+ +-v-XXX-v-+ +-v-XXX-v-+ +-----+ +-----+ +-----+ +-----+ +-----+
 |   | |         | |         | |Class| |Class| |Class| |Class| |Class|
 |CDT| | Class A | | Class B | | BE4 | | BE3 | | BE2 | | BE1 | | BE0 |
```

skipping to change at *page 17, line 31*

```
   |        |           |         |       |       |       |       |
 +-v--------v-----------v---------v-------V-------v-------v-------v--+
 |                     Strict Priority selection                     |
 +--------------------------------+----------------------------------+
                                  |
                                  V
```

Figure 4: The architecture of an output port inside a relay node with interleaved regulators (IRs) and credit-based shaper (CBS)

Each of the queuing subsystems for classes A and B, contains Credit-Based Shaper (CBS). 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 idleslope (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 (Section 8.6.8.2 of [IEEE8021Q]). 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 AVB traffic can be provided only if CDT traffic is bounded; it is assumed that the CDT traffic has leaky bucket arrival curve with two parameters r_h as rate and b_h as bucket size, i.e., the amount of bits entering a node within a time interval t is bounded by r_h * t + b_h.

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 DetNet flows traversing the network, which is a non-deterministic polynomial-time hard (NP-hard) problem [Sch8021Qbv]. Also, at this 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 the queuing model, we considered 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 together referred as AVB flows. 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

skipping to change at *page 16, line 46*

Shaper [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 queuing subsytem; there is no regulator for such classes.

I assume by "such classes" you mean CDT and BE flows? If so please be more clear, as in "there is no regulator for the latter" or clearer still just name them, as in "there is no regulator for CDT or BE flows".

```
       +--+   +--+ +--+   +--+
       |  |   |  | |  |   |  |
       |IR|   |IR| |IR|   |IR|
       |  |   |  | |  |   |  |
       +-++XXX++-+ +-++XXX++-+
         |     |     |     |
         |     |     |     |
 +---+ +-v-XXX-v-+ +-v-XXX-v-+ +-----+ +-----+ +-----+ +-----+ +-----+
 |   | |         | |         | |Class| |Class| |Class| |Class| |Class|
 |CDT| | Class A | | Class B | | BE4 | | BE3 | | BE2 | | BE1 | | BE0 |
```

skipping to change at *page 17, line 31*

```
   |        |           |         |       |       |       |       |
 +-v--------v-----------v---------v-------V-------v-------v-------v--+
 |                     Strict Priority selection                     |
 +--------------------------------+----------------------------------+
                                  |
                                  V
```

Figure 4: The architecture of an output port inside a relay node with interleaved regulators (IRs) and 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. The credit for each class A or B increases based on the idleslope (as guaranteed rate), and decreases based on the sendslope (typically equal to the difference

You use "idleslope" and "sendslope" here (with no space between words). Later in the document you use "idle slope" (my grep didn't find any instances of "send slope"; this is evidently the only time you reference it). Unless you have a reason to prefer the two different terms, please settle on one way of writing it (I prefer "idle slope" with the space).

between the guaranteed and the output link rates), both of which are parameters of the CBS (Section 8.6.8.2 of [IEEE8021Q]). 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 AVB traffic can be provided only if CDT traffic is bounded; it is assumed that the CDT traffic has a leaky bucket arrival curve with two parameters r_h as rate and b_h as bucket size, i.e., the amount of bits entering a node within a time interval t is bounded by r_h * t + b_h.

Additionally, it is assumed that the AVB flows are also regulated at their source according to leaky bucket arrival curve. At the source, the traffic satisfies its regulation constraint, i.e. the delay due to interleaved regulator at 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 of that all clocks are perfect. However, in reality there is clock nonideality thoughout 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 that, when configuring the regulators, the rates are set as the source rates with some positive margin. [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) for an AVB flow of classes A or B can be computed if the following condition holds:

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; 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 class 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; r_h as rate and b_h as bucket size of CDT traffic 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)

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

Then, an end-to-end delay bound of class X (A or B)is calculated by the formula Section 4.2.2, where for Cij:

Cij = d_X

Additionally, it is assumed that the AVB 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 nonideality thoughout 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 that, when configuring the regulators, the rates are set as the source rates with some positive margin. [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) for an AVB flow of classes A or B can be computed if the following condition holds:

I guess this section (or rather, this part of this section!) is talking about the delay bound on an individual node. I suppose that's obvious, ish, from the lead-in where you reference the queueing subsystem -- so the delay bound is in the context of a given node, which is where a given queueing subsystem is instantiated.

Nevertheless, it would have helped me if there had been some explicit text reminding the reader of this. My later comment will hopefully illuminate why this was problematic to my understanding.

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; 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 class 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

Is "idle slope" a well-known term of art? You do sort-of define it in §6.4, that might be sufficient in any case, especially once you rationalize "idleslope" vs. "idle slope".

length of class B and BE packets; L_n is the maximum packet length of classes A,B, and BE; r_h as rate and b_h as bucket size of CDT traffic leaky bucket arrival curve.

When you write "as rate" and "as bucket size" do you mean "is rate" and "is bucket size"? (If you really mean "as" that doesn't make sense to me.)

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; L_BE is the maximum packet length of class BE.

Then, an end-to-end delay bound of class X (A or B)is calculated by the formula from Section 4.2.2, where for Cij:

Cij = d_X

A few comments on the above. First, "the formula from Section 4.2.2" doesn't seem like a good cite, since what's presented in §4.2.2 is only an example, specific to a network of five nodes. I mean, it's probably not hard for the reader to extrapolate what you mean, but I think it's worth going to the small additional effort to state it as rigorously as the rest of your material.

Second, shouldn't you be parameterizing d_X in some fashion similar to the way you parameterize C? That is, if you are talking about Cij, then wouldn't you be talking about d_Xij? Pick your own terminology of course, or even explain it in prose, but the real point is that for the casual reader (me) it's not obvious until re-examination that d_X is specific to a given node.

In the end I THINK what you are saying here is along the lines of, the e2e delay bound for class X is given by the sum of the individual delay bounds d_X computed for each node along that path. Yes, no? (This is similar to how you write the delay bound in §6.5.)

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

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

6.4.2. Flow Admission

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 CBS for each class A and B (I_A and I_B), as well as the transmission rate of the output link (c). Besides, it is necessary to have the information on each class, i.e. maximum packet length of classes A, B, and BE. Moreover, the leaky bucket parameters of CDT (r_h,b_h) should be known. To admit a flow/flows of classes A and B, their

The delay bound calculation requires some information about each node. For each node, it is required to know the idle slope of CBS for each class A and B (I_A and I_B), as well as the transmission rate of the output link (c). Besides, it is necessary to have the information on each class, i.e. maximum packet length of classes A, B, and BE. Moreover, the leaky bucket parameters of CDT (r_h,b_h) should be known. To admit a flow/flows of classes A and B, their

I realize this isn't a Standards Track document and you're not using RFC 2119 language. Still, your use of "should" above, and following, is potentially a little problematic. I suggest you search through the document for instances of "should" and for each, consider whether you can replace it with "must" or something similarly unambiguous.

delay requirements should be guaranteed not to be violated. As described in Section 3.1, the two problems, static and dynamic, are addressed separately. In either of the problems, the rate and delay should be guaranteed. Thus,

The static admission control:

The leaky bucket parameters of all AVB flows are known, therefore, for each AVB flow f, a delay bound can be calculated. The computed delay bound for every AVB flow should not be more than its delay requirement. Moreover, the

delay requirements should be guaranteed not to be violated. As described in Section 3.1, the two problems, static and dynamic, are addressed separately. In either of the problems, the rate and delay should be guaranteed. Thus,

The static admission control:

The leaky bucket parameters of all AVB flows are known, therefore, for each AVB flow f, a delay bound can be calculated. The computed delay bound for every AVB flow should not be more than its delay requirement. Moreover, the

skipping to change at *page 20, line 31*

skipping to change at *page 20, line 31*

flow. Similarly, when an AVB flow leaves the network, all variables R_acc and b_acc along its path must be decremented to reflect the removal of the flow.

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

flow. Similarly, when an AVB flow leaves the network, all variables R_acc and b_acc along its path must be decremented to reflect the removal of the flow.

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

I've tried to be relaxed about your citation of the various IEEE documents as Informational, but I can't see how the above citation isn't Normative. There is no way for me to know what the constraints on R are without referring to the reference, that's the essence of Normative.

6.5. Guaranteed-Service IntServ

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

6.5. Guaranteed-Service IntServ

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

skipping to change at *page 21, line 38*

skipping to change at *page 21, line 38*

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 per-hop latency is trivially determined by the cycle time T_c: the packet transmitted from a node at a cycle (i), is transmitted from the next node at cycle (i+1). Hence, the maximum delay experienced by a given packet is from the beginning of cycle (i) to the end of cycle (i+1), or 2T_c; also, the minimum delay is from the end of cycle (i) to the beginning of cycle (i+1), i.e., zero. Then,

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 per-hop latency is trivially determined by the cycle time T_c: the packet transmitted from a node at a cycle (i), is transmitted from the next node at cycle (i+1). Hence, the maximum delay experienced by a given packet is from the beginning of cycle (i) to the end of cycle (i+1), or 2T_c; also, the minimum delay is from the end of cycle (i) to the beginning of cycle (i+1), i.e., zero. Then,

2T_c and zero as the maximum and minimum delays are relatively obvious. (Well, zero is not so obvious actually and I have doubts about it.) However, what follows is not so obvious:

if the packet traverses h hops, the maximum delay is:

(h+1) T_c

and the minimum delay is:

(h-1) T_c

if the packet traverses h hops, the maximum delay is:

(h+1) T_c

and the minimum delay is:

(h-1) T_c

which gives a latency variation of 2T_c.

The cycle length T_c should be carefully chosen; it needs to be large enough to accomodate 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 the delays 1,2,3,4 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 is 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

which gives a latency variation of 2T_c.

The naïve reader (me again) would assume that if a packet traverses h hops, and each hop has maximum delay 2T_c, then the per-path maximum delay would be the sum of the per-hop delays, i.e. (h) 2T_c. Likewise, I would have thought that if the minimum per-node delay is zero then the per-path minimum delay is trivially zero. (Presumably link latency isn't accounted for here, or else "zero" is just wrong.)

But this isn't what you say at all. Possibly there's something about CQF that provides the per-path property you state, e.g. something along the lines of, once I'm scheduled into a given cycle, the properties of the algorithm provide that I'll never encounter contention as I proceed through the network and thus at worst I'll endure 2T_c delay to be aligned to my cycle. But, this isn't stated, and your prose jumps without motivation from stating one set of properties for per-node delays to a surprising and non-obvious different set of properties for per-path delays. The "Then" in your prose implies that you've connected the two. AFAICT, you haven't.

The cycle length T_c should be carefully chosen; it needs to be large enough to accomodate 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 the delays 1,2,3,4 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 is 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

skipping to change at ***page 24, line 5***

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 Section 4.2.2 and Section 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 new set of valid paths and redo the delay bound computation or do not admit 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 [I-D.ietf-detnet-controller-plane-framework]).

8. Security considerations

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

skipping to change at ***page 23, line 51***

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 Section 4.2.2 and Section 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 new set of valid paths and redo the delay bound computation or do not admit the DetNet flow.

"do not admit" is ungrammatical as used. I suggest "reject" or "choose not to admit".

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 [I-D.ietf-detnet-controller-plane-framework]).

8. Security considerations

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

skipping to change at ***page 24, line 29***

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] that 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 a network node within the boundary of the DetNet domain.

A security consideration for this document is to secure the resource reservation signaling for DetNet flows. Any forge or manipulation of packets during reservation may lead the flow not to be admitted or face delay bound violation. Security mitigation for this issue is describedd in Section 7.6 of [RFC9055].

9. IANA considerations

This document has no IANA actions.

10. References

skipping to change at ***page 24, line 29***

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] that 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 a network node within the boundary of the DetNet domain.

A security consideration for this document is to secure the resource reservation signaling for DetNet flows. Any forgery or manipulation of packets during reservation may lead the flow not to be admitted or face delay bound violation. Security mitigation for this issue is describedd in Section 7.6 of [RFC9055].

9. IANA considerations

This document has no IANA actions.

10. References

skipping to change at ***page 26, line 36***

J.-Y. Le Boudec, "A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators", <https://ieeexplore.ieee.org/document/8519761>.

[NetCalBook]
J.-Y. Le Boudec and P. Thiran, "Network calculus: a theory of deterministic queuing systems for the internet", 2001, <https://ica1www.epfl.ch/PS_files/NetCal.htm>.

skipping to change at ***page 26, line 36***

J.-Y. Le Boudec, "A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators", <https://ieeexplore.ieee.org/document/8519761>.

[NetCalBook]
J.-Y. Le Boudec and P. Thiran, "Network calculus: a theory of deterministic queuing systems for the internet", 2001, <https://ica1www.epfl.ch/PS_files/NetCal.htm>.

	This URL has changed, it redirects to https://leboudec.github.io/netcal/
[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>.	[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>.	[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",	[RFC8578] Grossman, E., Ed., "Deterministic Networking Use Cases",

End of changes. 38 change blocks.

20 lines changed or deleted	*157 lines changed or added*

This html diff was produced by rfcdiff 1.48. The latest version is available from http://tools.ietf.org/tools/rfcdiff/