Deterministic Networking (detnet) Working Group
Internet-Draft
Intended status: Standards Track
Expires: 22 October 2025

A. Fressancourt, Ed.
Huawei

20 April 2025

Enforcing end-to-end delay bounds via queue resizing
draft-aft-detnet-bound-delay-queue-04

Abstract

This document presents a distributed mechanism to enforce strict delay bounds for some network flows in large scale networks. It leverages on the capacity of modern network devices to adapt their queue’s capacities to bound the maximum time spent by packets in those devices. It is using a reservation protocol to guarantee the availability of the resources in the devices’ queues to serve packets belonging to specific flows while enforcing an end-to-end delay constraint.

Status of This Memo

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

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

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

This Internet-Draft will expire on 22 October 2025.

Copyright Notice

Table of Contents

1. Introduction

While constraining both the latency and the jitter makes sense for mission-critical real-time applications, some applications can accommodate a relaxation of the jitter constraint provided a bound on the end-to-end latency is respected in order to operate over longer distances. Such use cases comprise online multiplayer gaming, augmented reality (AR) or virtual reality (VR), presented in [TS23501], as well as synchronized stream playback or wide-area monitoring and control systems, presented in [RFC8578]. In those use cases, packets may be buffered and reordered at the receiving end provided the data they carry can be delivered on time to the application.

This document presents a networked system designed to enforce strict delay bounds for some network flows in large scale networks. It is using both a dynamic mechanism to adapt queue capacity in network devices and a distributed signaling mechanism.

In this system, the network devices have a set of queues which are served for a amount of time on a regular period, using a round robin strategy or one of its variants. By controlling the amount of data packets that can be stored in each queue, the devices can control the maximum amount of time spent by packets in each queue, and commit on a maximum time spent to route packets belonging to a given flow. As each device can adjust the size of its various queues, the network can adapt to the demand for latency-bound traffic.

Then, in a network connecting such devices, it becomes possible to build end-to-end paths on which the maximum delay is bounded. A distributed signaling protocol is used to allow end devices to reserve capacity slots in the network devices queues in order to send traffic respecting end-to-end delay bounds to other end devices.

2. Bounding delay at the switches

[RFC9320] gives a detailed description of the timing model for relay nodes involved in a deterministic network. In this document, using the same notations and numbering, the transit time from a given node to its successor on the path is the sum of:

1. An output delay.

2. A link delay;

3. A frame preemption delay (according to [IEEE8023]);

4. A processing delay;

5. A regulator queueing delay;

6. A queueing subsystem delay;

Components 1 to 4 in this per hop delay computation can be bounded by a non-queuing delay upper bound, while components 5 and 6 constitute a queuing delay depending on the queuing strategy.

In modern network equipment, several queues are used to park packets depending on a set of criteria. In some architecture, such as the Protocol Independent Switch Architecture (see Figure 1), the queues are located before the egress processing pipeline. The time spent by a packet in an equipment depends on the time spent in the processing

pipelines, on the time it stays in a queue and on potential packet reordering in the queue. Considering a round robin service of those queues, if no packet reordering operation is done in the queues, the time spent by packets in a queue depends on the queue's capacity.

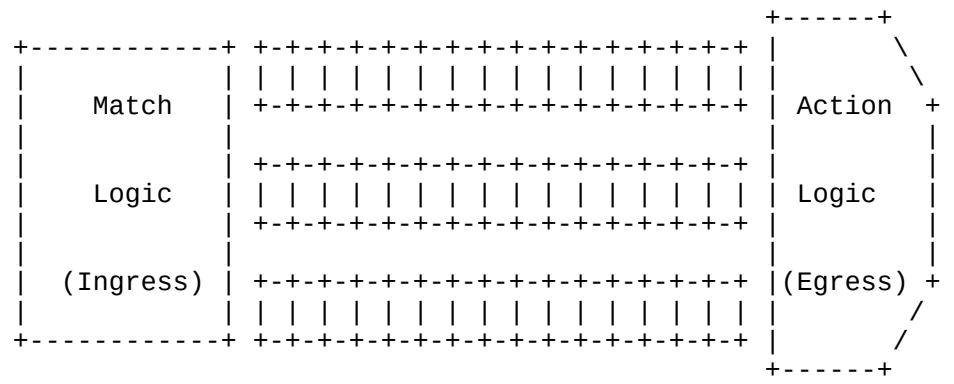


Figure 1: A schematic view of the PISA architecture

If the queues are served as FIFO, then, according to [LeBoudecTheory], the worst node delay, i.e. the maximum time spent by a packet in transit (M.D.) in a given node can be expressed as:

M.D. = T0 + Cp / CIR

Where:

* T0 is the upper bound of the non-queuing delay experienced by a packet in the node,

* Cp is the buffer capacity (in bits),

* CIR is the Committed Information Rate, i.e. the service rate of the queue (in bits per second (bps)).

According to this formula, it is possible to bound the time spent by a packet in an equipment by assigning it to a queue of the proper size. Besides, the maximum time spent by packets in the equipment can be adjusted by changing the buffer size, i.e. the number of packets in the queue. Thus, if each packet's size is bounded by the MTU, it is possible to express the buffer sizes as capacity reserved for specific network flows.

In the described system, end nodes commit to respect a strict threshold on the bandwidth they consume for data flows for which they want the end-to-end delay to be bound. Indeed, hosts are allowed to

send bursts of traffic violating the maximum bandwidth they are allowed to consume, the maximum delay experienced by packets in the various network nodes will not be bounded, as space in the various queues might be illegitimately allocated. To respect this constraint, end hosts are forced to strictly respect the threshold they are allocated for their delay-bound flow, and reject every packet violating this cap at the network's ingress.

In this system, the capacity of the various queues in the equipment is steered, and end host can reserve capacity in those queues by means of a reservation protocol able to carry queue reservation parameters in reservation request and reply messages.

3. Setting up an end-to-end path with a delay bound

To describe how a path with strict delay bounds can be set up, let us consider as an example the network presented in Figure 2. This network consists in a source node A, a destination node F, and four (4) intermediate nodes A, B, C and D. Each of these four intermediate nodes have 2 guaranteed service queues, characterized by the Maximum node transit Delay they offer (M.D, in ms) and their capacity (Cp, in Mbits), and a best effort queue. Those queues are served in a round robin fashion, in such a way that their service time is guaranteed. In this network, A wants to send traffic to F in a data flow, with a maximum capacity of 2Mbits and a end-to-end delay limit of 85 ms.

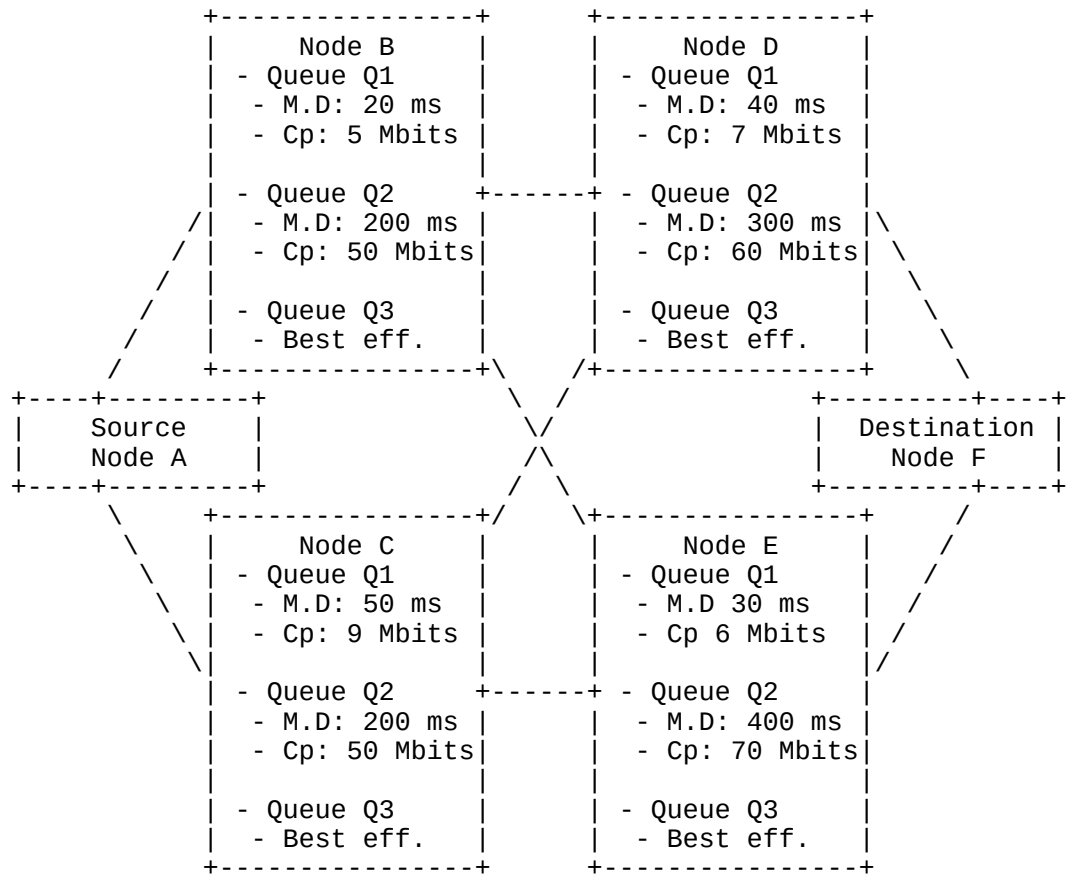


Figure 2: Example network

To set the path, A will use the resource reservation protocol to send a reservation request to F carrying the end-to-end delay bound and the maximum capacity of the data flow it is willing to send. This message is denoted Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 0ms; M.C. = 2Mbits; Record-route = [A]]. It contains six (6) parameters of interest for the reservation procedure:

* a nonce ID (ID) that identifies the request;

* a flow ID (Flow-ID) which follows the requirements stated in Section 5.1 of [RFC8939] to identify the flow for which the resource reservation is done;

* a maximum end-to-end delay (max-E2E-delay) to be respected on the path from the source to the destination;

* an end-to-end delay commitment (E2E-delay-commit.) which represents the sum of the maximum delays that nodes traversed by the packet commit to respect;

* a maximum capacity (M.C.) which is the upper bound of the capacity required to serve the data flow on the path;

* a recorded route (Record-route) that is used to determine which nodes the message has traversed.

The message is sent to B and C.

When B receives the message, it checks that it can participate in the path by checking whether it can accept packets with the requested capacity in one of its queues while respecting the end-to-end delay. It can place packets associated to this flow in its queue Q1. Then, before forwarding the packet, it sets a temporary reservation in its queue Q1, adds Q1's maximum delay (20ms) to the end-to-end delay commitment in the reservation request message, and adds itself in the Record-route stack in the message: Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 20ms; M.C. = 2Mbits; Record-route = [A, B]]. Then it relays the packet to D and E.

At the same time, C proceeds in a similar way, assigns a temporary reservation to its queue Q1 and sends messages Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 50ms; M.C. = 2Mbits; Record-route = [A, C]] to D and E.

When D receives both messages Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 20ms; M.C. = 2Mbits; Record-route = [A, B]] and Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 50ms; M.C. = 2Mbits; Record-route = [A, C]], it knows from the nonce ID that they are associated to the same request. Looking at the first message, it determines that it can serve this flow by reserving some capacity in its queue Q1. It thus creates a message Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 60ms; M.C. = 2Mbits; Record-route = [A, B, D]] by adding Q1's maximum delay to the end-to-end delay commitment and recording itself in the Record-Route, and forwards it to F. Yet, looking at the second message, it realizes that it cannot answer the request given that the maximum delay of its fastest queue is higher than the end-to-end delay value requested in the message. Then, it silently refrains from relaying the message.

At the same time, E processes both messages Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 20ms; M.C. = 2Mbits; Record-route = [A, B]] and Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 50ms; M.C. = 2Mbits; Record-route = [A, C]], makes a temporary reservation in its queue Q1 and relays messages Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 50ms; M.C. = 2Mbits; Record-route = [A, B, E]] and Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 80ms; M.C. = 2Mbits; Record-route = [A, C, E]] to F.

F, the destination, receives 3 messages dealing with the same end-to-end resource reservation request from A:

* Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 60ms; M.C. = 2Mbits; Record-route = [A, B, D]];

* Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 50ms; M.C. = 2Mbits; Record-route = [A, B, E]];

* Req.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 80ms; M.C. = 2Mbits; Record-route = [A, C, E]].

From this set, F chooses a path. The specific policy used to make a selection among several possible paths is out of the scope of this document. In this example, F selects the path on which the end-to-end delay commitment value is the lowest, and answers the resource reservation request with a response containing six (6) parameters:

* the nonce ID that identifies the request;

* the identifier of the flow for which the resource reservation is done;

* the maximum end-to-end delay to be respected on the path from the source to the destination;

* a delay contract value, which is the end-to-end delay commitment received by the destination on the path;

* a capacity;

* a route that is used to determine which nodes the message has traversed.

F sets the route to [A, B, E, F] in its response, and sets the delay contract to 50ms, and sends to E the message Res.[ID = Nonce; Flow-ID = f-ID; max-E2E-delay = 85ms; E2E-delay-commit. = 50ms; M.C. = 2Mbits; Route = [A, B, E, F]]. This response is relayed to A through

E and B following the information in the Route parameter (see for instance the behavior presented in Section 5.4 and Section 5.5). When they receive the message, both E and B confirm the temporary reservation they set for the request in their Q1 queues. When A receives the response, it knows that an end-to-end path going through B and E to reach F has been set, and that it respects the end-to-end delay constraint.

4. Adapting the queues capacities

In the lifetime of a network, the demand for delay-bound traffic can change: the distribution of the delay bound requests can shift, requiring the nodes to adapt the size of their various buffers.

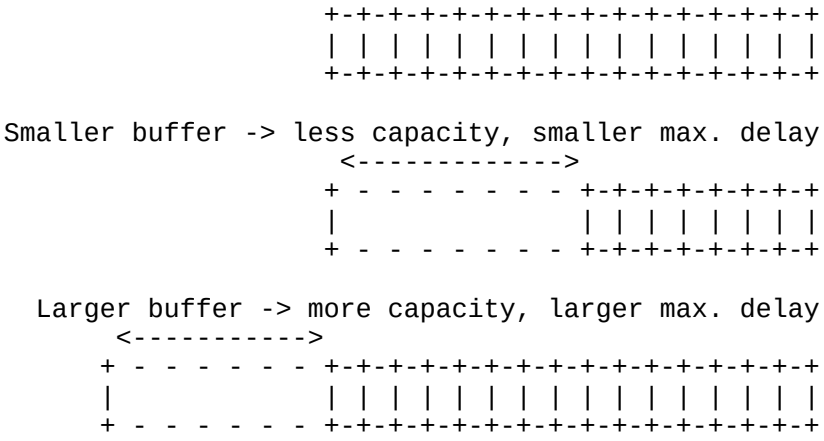


Figure 3: Influence of buffer size on capacity and maximum delay

A node participating in this system maintains a list of the currently active reservations in its queues, associated with their characteristics (i.e. the flow identifier, the requested capacities and the delay bound commitment to respect at the node).

The queues' capacities are not fixed, and can evolve with time. In the reservation protocol's operations, two types of events can lead a node to adapt the capacity of one of its queues:

* If a node receives a reservation request it can not serve because the requested delay is too low, and the occupation of one of its queues is below an occupation threshold (20% of the current queue capacity for instance), then the node can reduce the buffer size to be able to enforce a lower delay for flows assigned to the queue and thus accept the request (see middle of Figure 3).

* If a node receives a reservation request it can not serve because the requested capacity is too high, then the node can observe the occupation and the minimum delay of active flow reservations in its queues. If one of its queues' minimal delay is under the minimum active flow delay commitment, the node can look whether augmenting the queue's delay to meet this minimum commitment can help it accept the request (see bottom of Figure 3). In a proactive way, the node can set an occupation threshold (80 % for instance) above which it inspects the minimum delay commitment for active flows in a queue to see whether it can enlarge the queue's size to accept more flows.

Those buffer size adaptation operations can be performed independently by each node. Indeed, as presented in Section 3, when a source reserves resources to respect an end-to-end latency bound, each intermediate node takes a local commitment about a per node maximum delay it aims at respecting. If, at a given time, the Maximum node transit Delay of a queue is smaller that the minimum delay commitment for active reservations in this queue, then the node can adapt this queue's size without interfering with any of the commitments it has taken. Thus, as long as the queue size adaptation respects ongoing commitments, this procedure do not require additional signalling.

5. End-to-end ressource reservation protocol operations

5.1. Using the RSVP protocol

The RSVP protocol is used to allow an end device to reserve a path in a network consisting in devices able to steer their queues depth. Both the RSVP Path message and the RSVP Resv message are exchanged to set a queue capacity reservation on a path and acknowledge it.

To reserve capacity to serve a flow respecting a delay bound, the Path message carries a set of objects to carry the request information presented in the example described in Section 3. The acknowledgement of the resource reservation is made by using a Resv message. It carries the request information presented in the example described in Section 3.

5.2. Information encoding in RSVP messages

The RSVP messages used in the protocol's operations need to convey the information listed in Section 5.1. Previous RFCs, namely [RFC2210], [RFC2212] and [RFC2215], present data objects that can be used to carry this information.

According to [RFC2210], it is needed to use RSVP to agree on a guaranteed service to enforce a latency bound between the source and destination. This will influence the data provided in the various objects carried by both the Path and Resv messages.

The reservation ID and the identifier associated with the flow for which the reservation is done are carried in the SESSION data object. [RFC2205] mentions that in the RSVP SESSION object, the flow is identified by the destination address and optionally by the destination port, the protocol ID and a 1-byte flags field. This limits the number of Detnet flows that can be identified compared to the requirements of [RFC8939] Section 5.1. Route and record route elements lists are carried by EXPLICIT_ROUTE and ROUTE_RECORD objects (see [RFC3209]).

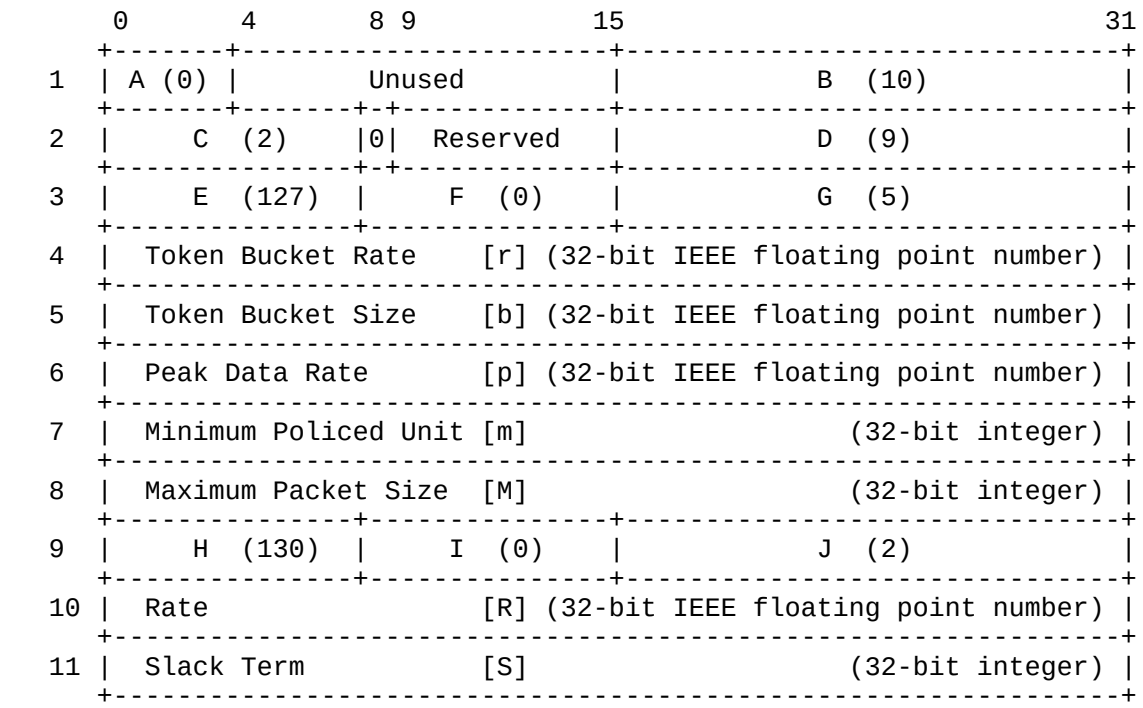
The rest of the information carried by the Path and Resv messages deal with the characterization of the data flow. According to [RFC2210], the end to end QoS characteristics of the flow should be carried either by a SENDER_TSPEC object in the Path message or by a FLOWSPEC object in the Resv message. The information given by either the source or destination in those objects is not changed by on path nodes. For the reservation procedure described in this document, it means that the maximum end-to-end delay and the Capacity values need to be conveyed by those objects. On the contrary, the end-to-end delay commitment value needs to be modified by on path nodes. To carry this piece of information, both RSVP Path and Resv message can use an ADSPEC object.

According to [RFC2210], the SENDER_TSPEC and the FLOWSPEC objects need to include both a general token bucket TSpec parameter and a guaranteed service RSpec parameter. Those parameters consist in words characterizing the desired properties of the end-to-end flow. The token bucket TSPEC contains five (5) words: (1) a token bucket rate r, (2) a token bucket size b, (3) a peak data rate p, (4) a minimum policed unit m and (5) a maximum packet size M. The guaranteed service RSpec contains two (2) words: (1) a rate R and (2) a slack term S. Those parameters' words are used to characterized the desired end-to-end delay bound (Dreq). According to [RFC2212], it is given by the formula:

$$Dreq = S + b/r$$

when p=r=R in the parameters carried by the SENDER_TSPEC and the FLOWSPEC objects.

From this formula, a sender node can formalize its end to end delay contract by computing b/r and substracting this value from the desired end-to-end delay to obtain the value of the slack term S to be added as a word in the guaranteed service RSpec.



- A : Message format version number (0)
- B : Overall length (9 words excluding header)
- C : Service header, service number 2 (Guaranteed)
- D : Per-service data length, (9 words excluding per-service header)
- E : Parameter ID, parameter 127 (Token Bucket TSpec)
- F : Parameter 127 flags (none set)
- G : Parameter 127 length, 5 words excluding parameter header
- H : Parameter ID, parameter 130 (Guaranteed Service RSpec)
- I : Parameter 130 flags (none set)
- J : Parameter 130 length, 2 words excluding parameter header

Figure 4: Token bucket TSpec and guaranteed service RSpec parameters carried by the SENDER_TSPEC and the FLOWSPEC objects

The ADSPEC object carried by the Path and the Resv messages needs to include a set of default general parameters as well as a fragment carrying guaranteed service parameters. There are five (5) default parameters to include: (1) The global break bit, (2) the IS hop

count, (3) the path bandwidth, (4) the minimum path latency and (5) the composed path MTU. The guaranteed service fragment needs to include four (4) parameters: (1) the end-to-end composed value of the rate-dependent error term Ctot, (2) the end-to-end composed value of the rate-independent error term Dtot, (3) the value of the rate-dependent error term since the last composition point Csum and (4) the value of the rate-independent error term since the last composition point Dsum. In the reservation procedure, the minimum path latency is set to the undetermined value specified in [RFC2215], i.e. (2**32)-1, to signal that the propagation delay is not considered. The on path node provide their contribution to the end-to-end delay commitment by adding the delay bound they commit to respect to both the Dtot and the Dsum parameters, while keeping the Ctot and the Csum parameters with a zero value. Other parameters appearing in the ADSPEC object are set and modified according to the procedure described in [RFC2210].

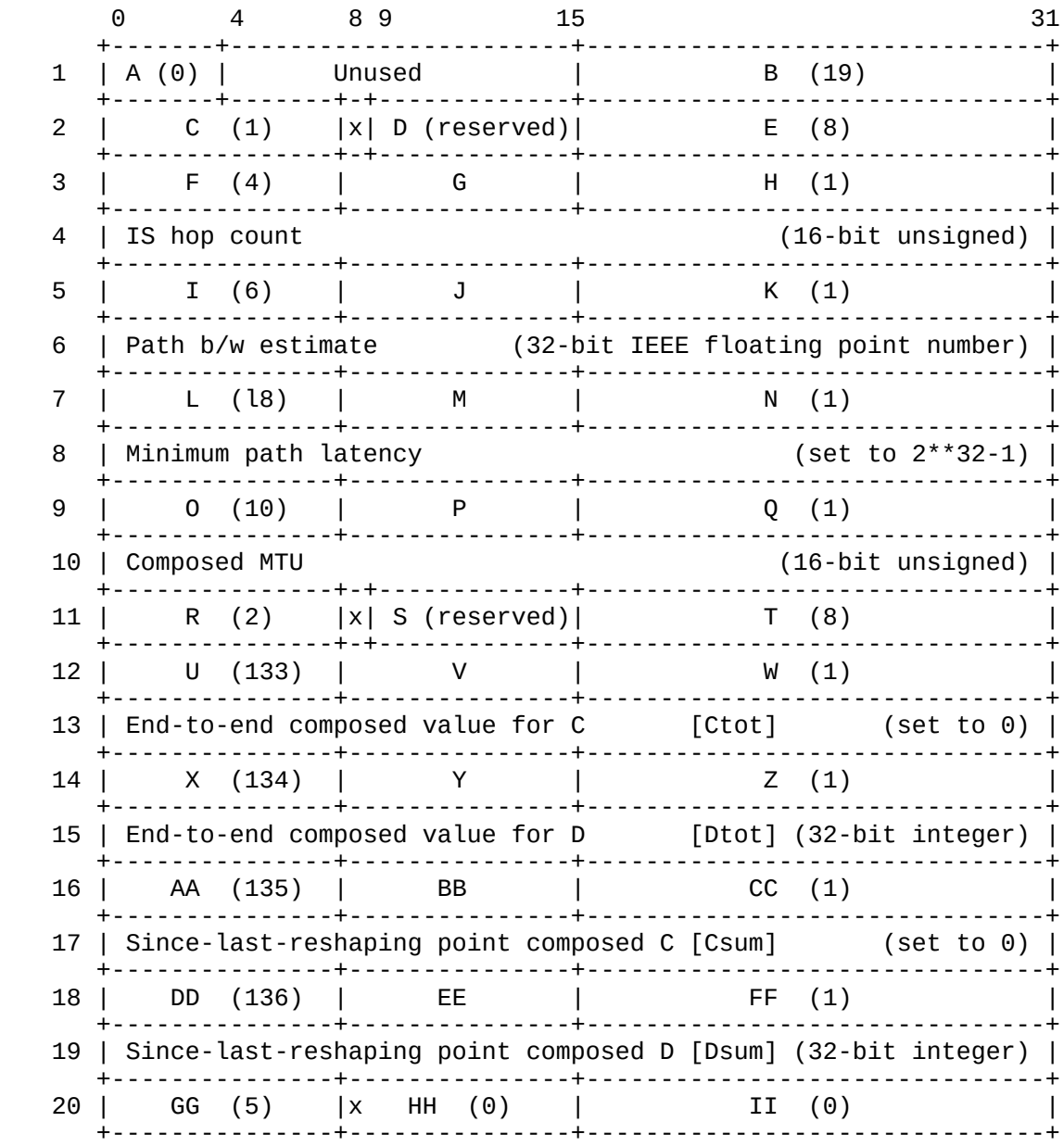


Figure 5: ADSPEC object format

```
   Word 1: Message Header:
   - A : Message header and version number
   - B : Message length (19 words excluding header)

   Words 2-7: Default general characterization parameters
   - C : Per-Service header, service number 1
         (Default General Parameters)
   - D : Global Break bit (NON_IS_HOP general parameter 2) (marked x)
   - E : Length of General Parameters data block (8 words)
   - F : Parameter ID, parameter 4 (NUMBER_OF_IS_HOPS general parameter)
   - G : Parameter 4 flag byte
   - H : Parameter 4 length (1 word excluding header)
   - I : Parameter ID, parameter 6
         (AVAILABLE_PATH_BANDWIDTH general parameter)
   - J : Parameter 6 flag byte
   - K : Parameter 6 length (1 word excluding header)
   - L : Parameter ID, parameter 8
         (MINIMUM_PATH_LATENCY general parameter)
   - M : Parameter 8 flag byte
   - N : Parameter 8 length (1 word excluding header)
   - O : Parameter ID, parameter 10 (PATH_MTU general parameter)
   - P : Parameter 10 flag byte
   - Q : Parameter 10 length (1 word excluding header)

   Words 11-19: Guaranteed service parameters
   - R : Per-Service header, service number 2 (Guaranteed)
   - S : Break bit
   - T : Length of per-service data (8 words excluding header)
   - U : Parameter ID, parameter 133 (Composed Ctot)
   - V : Composed Ctot flag byte
   - W : Composed Ctot length (1 word excluding header)
   - X : Parameter ID, parameter 134 (Composed Dtot)
   - Y : Composed Dtot flag byte
   - Z : Composed Dtot length (1 word excluding header)
   - AA: Parameter ID, parameter 135 (Composed Csum).
   - BB: Composed Csum flag byte
   - CC: Composed Csum length (1 word excluding header)
   - DD: Parameter ID, parameter 136 (Composed Dsum).
   - EE: Composed Dsum flag byte
   - FF: Composed Dsum length (1 word excluding header)

   Word 20: Controlled-Load parameters
   - GG: Per-Service header, service number 5 (Controlled-Load)
   - HH: Break bit
   - II: Length of controlled-load data (0 words excluding header)
```

Figure 6: Caption for ADSPEC object format

5.3. Operations at the source

The source is the end host willing to send traffic to a destination in a data flow with a delay guarantee. To do so, it sends a RSVP Path message to the destination. This RSVP Path message encodes the information listed in the reservation request described in Section 3 according to the encoding method described in Section 5.1. The Path message triggers a RSVP Resv message that answers to the according delay-bound path request. At the reception of a Resv message replying to a pending request, the source waits for a bit in order to give time to other potential answers to arrive. If a single answer is received, then the source starts using the path received in the EXPLICIT_ROUTE object of the Resv message. If multiple answers have been received, then the source chooses a path, and sends a RSVP PathTear message to the paths that have not been selected.

5.4. Operations at intermediate nodes

At the reception of a Path message carrying an end-to-end delay request, it first checks that the message is not duplicated by looking at its ID. Then, it looks at both the Delay and Capacity values, and determines whether it can accept the request or not.

If it can accept the request, then the intermediate node relays the RSVP Path message to the destination. Before relaying the message, it substracts the maximum delay he commits to respect from the end-to-end delay, and adds its identifier in the ROUTE_RECORD object. If it knows several paths to the destination, it can duplicate the message and relay it on the appropriate egress paths.

At the reception of a Resv message carrying a reply to an end-to-end delay request, it first checks that the message is not duplicated. Then, it verifies that it has a pending temporary capacity reservation associated with the reply for one of its queues. If it is the case, then it acknowledges the reservation, and allocates the dedicated capacity to the data flow. It check the next hop for the Resv message in the EXPLICIT_ROUTE object carried by the message, and relays it to the next node. If several Resv messages for the same data flow arrive at the intermediate node, the intermediate node relays all of them, as they might refer to different paths in the network from which the source end host needs to choose from. Temporary reservations following the reception of a Path message that are not confirmed by a Resv message are cancelled by the reception of a PathErr message for this flow or by the expiration of a timer set to one RSVP timeout period. Confirmed reservations may be teared down by a PathTear message or by the expiration of a cleanout timer set to the value of the RSVP cleanout period.

5.5. Operations at the destination

The destination node is in charge of building the RSVP Resv message from the RSVP path message it received from the source.

In case several Path messages have been sent or duplicated along the path to the destination, the destination can behave in two ways: either it chooses which path is the most appropriate from a set of Path message containing the same originating end-to-end delay request, or it replies to each request message and lets the source choose which path it will use.

If the destination is responsible for the choice of the path, then when it receives a Path message carrying an end-to-end delay request, it waits for a bit in order to give time to potential Path messages associated to the same request to arrive. If several Path messages have been received, the destination chooses one of the Path message (for instance the message carrying the highest Delay value, or the one carrying the shortest path in terms of hops), and forges a RSVP Resv message to answer this request. This Resv message carries the requested maximum end-to-end delay, the final end-to-end delay commitment received in the Path message, and a Route list containing the Record-route list received in the Path message to which the destination’s ID is added at the end. It also sends a set of PathErr messages for the Path messages that have not been selected to withdraw the reservation at the involved nodes.

If the destination lets the source choose, then, for each Path message carrying an end-to-end delay request object, the destination creates a Resv message in which the requested maximum end-to-end delay and the final end-to-end delay commitment are set to the values received in the Path message, and in which the Route list is set to the Record-route list received in the request message, with the destination’s identifier added at the end of the list.

Following [RFC3209], if the Path message received by the destination contains a ROUTE_RECORD object, then the destination adds an EXPLICIT_ROUTE object to the Resv message giving the list of nodes that have been crossed by the Path message. This EXPLICIT_ROUTE object tells intermediate nodes how they need to forward the Resv message.

6. Positioning in the Dataplane Enhancement Taxonomy

[I-D.joung-detnet-taxonomy-dataplane] presents a taxonomy of dataplane enhancements proposed in the Detnet working group. The mechanism presented in this document targets the same goals as those proposals. Here is how this work can be positionned.

* *Per Hop Dominant Factor for Latency Bound: Category 3.* In category 3, the per hop dominant factor is the sum of Max Burst Sizes/Capacity: the mechanism presented in this draft assumes strict control of the maximum capacity for flows it deals with, and focuses on the repect of a maximum latency bound in this regard.

* *Periodicity: Non-periodic.* The proposed mechanism does not assume packets are transmitted in a periodic pattern.

* *Network Synchronization: Asynchronous.* No synchronization between nodes is assumed in the presented mechanism.

* *Traffic Granularity: Flow level.* In the mechanism presented in this document, each packet is controlled based on its specific flow.

* *Work Conserving: Yes.* Queues in equipments are served in a round robin fashion, whatever their length.

* *Target Transmission Time: In-time.* The mechanism presented in this document aims at respecting a bounded end to end delay, whatever the jitter.

* *Service Order: Rate-based.* In the mechanism described in this document, packets belonging to a flow are assigned to queues depending on the per hop delay bound that the node has commited to respect for packets belonging to this specific flow.

7. Security Considerations

A detailled analysis of the security aspects of the current draft will be presented in a future version of the draft. Yet, the current document is not adding additional threats to the ones identified for RSVP and presented in Section 2.8 of [RFC2205].

8. IANA Considerations

This document has no IANA actions.

9. References

9.1. Normative References

[I-D.joung-detnet-taxonomy-dataplane]
Joung, J., Geng, X., Peng, S., and T. T. Eckert, "Dataplane Enhancement Taxonomy", Work in Progress, Internet-Draft, draft-joung-detnet-taxonomy-dataplane-01, 25 February 2024, <https://datatracker.ietf.org/doc/html/draft-joung-detnet-taxonomy-dataplane-01>.

[LeBoudecTheory]
Le Boudec, J. and P. Thiran, "A Theory of Deterministic Queuing Systems for the Internet", 2001.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <https://www.rfc-editor.org/rfc/rfc2205>.

[RFC2210] Wroclawski, J., "The Use of RSVP with IETF Integrated Services", RFC 2210, DOI 10.17487/RFC2210, September 1997, <https://www.rfc-editor.org/rfc/rfc2210>.

[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/rfc/rfc2212>.

[RFC2215] Shenker, S. and J. Wroclawski, "General Characterization Parameters for Integrated Service Network Elements", RFC 2215, DOI 10.17487/RFC2215, September 1997, <https://www.rfc-editor.org/rfc/rfc2215>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/rfc/rfc3209>.

[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/rfc/rfc8939>.

[RFC9320] Finn, N., Le Boudec, J.-Y., Mohammadpour, E., Zhang, J., and B. Varga, "Deterministic Networking (DetNet) Bounded Latency", RFC 9320, DOI 10.17487/RFC9320, November 2022, <https://www.rfc-editor.org/rfc/rfc9320>.

9.2. Informative References

[IEEE8023] "IEEE Standard for Ethernet", IEEE,
DOI 10.1109/ieeestd.2022.9844436, ISBN ["9781504487252"],
July 2022, <https://doi.org/10.1109/ieeestd.2022.9844436>.

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

[TS23501] 3rd Generation Partnership Project and D. Chandramouli,
"System architecture for the 5G System (5GS)",
<https://www.3gpp.org/ftp/Specs/
archive/23_series/23.501/23501-i00.zip>.

Appendix A. Acknowledgments

Thanks to Shaofu Peng for his remarks on the previous versions of this draft, that contributed in enhancing this document.

Contributors

Paolo Medagliani
Huawei Technologies France S.A.S.U.
Email: paolo.medagliani@huawei.com

Sebastien Martin
Huawei Technologies France S.A.S.U.
Email: sebastien.martin@huawei.com

Anne Bouillard
テ営ole Normale Supテゥrieure - PSL
Email: anne.bouillard@ens.fr

Author’s Address

Antoine Fressancourt (editor)
Huawei Technologies France S.A.S.U.
18, Quai du Point du Jour
92100 Boulogne-Billancourt
France
Email: antoine.fressancourt@huawei.com