DetNet Working Group
Internet-Draft
Intended status: Standards Track
Expires: 28 June 2025

J. Joung
Sangmyung University
J. Ryoo
T. Cheung
ETRI
Y. Li
Huawei
P. Liu
China Mobile

25 December 2024

Latency Guarantee with Stateless Fair Queuing
draft-joung-detnet-stateless-fair-queuing-04

Abstract

This document specifies the framework and the operational procedure for deterministic networking with a set of rate based work conserving packet schedulers. The framework guarantees end-to-end (E2E) latency bounds to flows. The schedulers in core nodes do not need to maintain flow states. Instead, the entrance node of a flow marks an ideal service completion time according to a fluid model, called Finish Time (FT), of a packet in the packet header. The subsequent core nodes update FT by adding delay factors, which are functions of the flow and the nodes. The packets in the queue of the scheduler are served in the ascending order of FT. This mechanism is called the stateless fair queuing. The result is that flows are isolated from each other almost perfectly. The latency bound of a flow depends only on the flow's intrinsic parameters such as the maximum burst size and the service rate, except the link capacities and the maximum packet length among other flows sharing each output link with the flow.

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 28 June 2025.

Table of Contents

1. Introduction

There are emerging applications that require both latency and jitter bounds in large-scale networks. One of the key mechanisms to the latency and jitter performance of a network is the packet scheduling in the data plane. Our objective is to specify a scheduling mechanism that can completely isolate a flow from other flows that are sharing a link. An ideal flow isolation would be achieved, if an imaginary link is dedicated solely to the flow across the network, with the capacity equal to the allocated service rate to the flow. In this case the latency upper bound is a function of the flow's parameters only, including the maximum burst size, maximum packet length, and the service rate.

In large-scale networks, end nodes can join and leave, and a large number of flows are dynamically generated and terminated. Achieving satisfactory deterministic performance in such environments would be challenging. The current Internet, which has adopted the differentiated services (DiffServ) architecture, has the problem of the burst accumulation and the cyclic dependency, which is mainly due to FIFO queuing and strict priority scheduling. Cyclic dependency is defined as a situation wherein the graph of interference between flow paths has cycles [THOMAS]. The existence of such cyclic dependencies makes the proof of determinism a much more challenging issue and can lead to system instability, that is, unbounded delays [ANDREWS][BOUILLARD].

A class of schedulers called Fair Queuing (FQ) limits interference between flows to the degree of maximum packet size. Packetized generalized processor sharing (PGPS) and weighted fair queuing (WFQ) are representative examples of FQ [PAREKH]. In FQ, the ideal service completion time, called Finish Time (FT), of a packet is obtained from an imaginary system which can provide the ideal flow isolation. Packets in the buffer are served in an increasing order of the FT. When this mechanism is applied, the end-to-end (E2E) latency bound of a flow is similar to that in the ideally isolated system. However, the FT of the previous packet within a flow has to be remembered for the calculation of the current packet's FT. This information can be seen as the flow state. The complexity of managing such information for a large number of flows can be a burden, so the FQ has not been usually adopted in practice.

This document specifies a framework for realizing the FQ scheduler in core nodes, without maintaining flow state. It also specifies the operational procedure based on this framework. The edge node through which a flow enters a network is called the entrance node. The entrance node for a flow generates FT for a packet and records it in the packet. A core node, based on these records, updates FT by

adding a delay factor that is a function of parameters of the node and the flow. The proposed framework is called work conserving stateless core fair queuing (C-SCORE) [C-SCORE]. C-SCORE is work conserving and has the property that, for a certain choice of the delay factor, the expression for E2E latency bound can be found. This E2E latency bound function is the same as that of a network with stateful FQ schedulers in all the nodes.

The key component of C-SCORE is the packet state that is carried as metadata. C-SCORE does not need to maintain flow states at core nodes, yet it works as one of the FQ schedulers, which is known to provide the best flow isolation performance. The metadata to be carried in the packet header is simple and can be updated during the stay in the queue or before joining the queue.

2. Terminology

2.1. Terms Used in This Document

2.2. Abbreviations

3. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. Fair Queuing Schedulers

Generalized processor sharing (GPS) suggested a paradigm for a fair service for flows as fluid. Packetized GPS (PGPS), which implemented GPS in the realistic packet-based environment, played a pioneering role in this type of packet-based schedulers [PAREKH]. PGPS determines the service order of packets in ascending order of the FT derived by the following equation.

$$F(p) = \max\{F(p-1), V(A(p))\}+L(p)/r, \quad (1)$$

where p and p-1 are the pth and (p-1)th packets of a flow, F(p) is the FT, A(p) is the arrival time, L(p) is the length of the packet p, and r is the service rate allocated to the flow. Note that the index for the flow i is omitted. V(t) is called the virtual time function [PAREKH]. and is a value representing the current system progress at time t. If the backlogged flows almost fill the link capacity, then

the system slowly progresses in terms of a flow's view, and the virtual time increases slowly. If there is only a handful of backlogged flows, then the virtual time increases with a higher rate. This behavior of the virtual time function prevents an unfair situation in which flows that entered late have relatively small FTs thus receive services earlier for a considerable duration of time, compared to existing flows.

F(p) represents the time that an ideal fluid system would complete its service of packet p. In a real packetized FQ system, the packets are served in the increasing order of the FT. The FT can be calculated at the moment the packet arrives in the node, and the value of FT is attached to the packet before the packet is stored in the buffer. In general, there is a queue for each flow, the queues are managed by FIFO manner, and the scheduler serves the queue having the HoQ packet with the smallest FT. Alternatively, it is possible to put all the packets in one queue and sort them, as packets are enqueued or dequeued, according to the value of the FT. This implementation requires a priority queue. The point of (1) is that, in the worst case, when all the flows are active and the link is fully used, a flow is served at an interval of L(p)/r. At the same time, by using the work conserving scheduler, any excessive link resources are shared among the flows. How fairly it is shared is the main difference between various FQ schedulers.

In order to obtain F(p) in (1), the FT of the previous packet of the flow, F(p-1), must be remembered. When a packet is received, it is necessary to find out which flow it belongs to and find out the FT of the latest packet of the corresponding flow. F(p-1) of this latest packet is a value representative of the 'flow state'. The fact that such state information must be memorized and read means a considerable complexity at a core node managing millions of flows. This is the main reason why such FQ schedulers are not actually used on the Internet. In this document, we pay attention to the fact that the fair service time interval information between packets is already included in the FTs of the entrance node of a flow. Instead of deriving a new FT at each core node, we will specify a method of deriving the FT at downstream nodes by using the initial FT calculated at the entrance node.

On the other hand, calculating V(t) also involves the complexity of calculating the sum of r by tracking the flows currently being serviced in real time. It is a complex calculation. Therefore, instead of calculating V(t) accurately, methods for estimating it in a simple way have been suggested. Among them, Virtual Clock (VC) [ZHANG] uses the current time t instead of V(t) to determine the FT. Self-clocked fair queuing [GOLESTANI] uses the FT of the recently serviced packet of another flow instead of V(t). The document adopts the VC's approach.

Stiliadis showed that this series of FQ schedulers can belong to the Packetized Rate Proportional Servers (PRPS) [STILIADIS-RPS]. For example, PGPS and VC are PRPS, while self-clocked fair queuing is not. It was proved that a network with all the nodes having one of these PRPSs guarantee the E2E latency for flow. Moreover, any PRPS scheduler yields the same E2E latency bound [STILIADIS-RPS].

5. Assumptions

In this document, we assume there are only two classes of traffic. The high priority traffic requires guarantee on latency upper bounds. All the other traffic is considered to be the low priority traffic, and be completely preempted by the high priority traffic. High priority traffic is our only concern.

All the flows conform to their traffic specification (TSpec) parameters. In other words, with the maximum burst size Bi and the arrival rate ai, the accumulated arrival from flow i in any arbitrary time interval [t1, t2], t1 < t2, does not exceed Bi+(t2-t1)ai. An actual allocated service rate to a flow, ri, can be larger than or equal to the arrival rate of the flow. As it will be shown in (5) that, by adjusting the service rate to a flow, the E2E latency bound of the flow can be adjusted. Note that ri is used interchangeably with the symbol r to denote the service rate of a flow. Total allocated service rate to all the flows in a node does not exceed the link capacity of the node. These assumptions make the resource reservation and the admission control mandatory.

A node, or equivalently a server, means an output port module of a switching device.

The entrance node for a flow is the node located at the edge of a network, from which the flow enters into the network. A core node for a flow is a node in the network, which is traversed by the flow and is not the entrance node. Note that a single node can be both an entrance node to a flow and a core node for another flow.

A packet is considered arrived or serviced; when its last bit has arrived or left the node, respectively.

Propagation delays are neglected for the simplicity of representation. However, it can be easily incorporated into the equations presented in this document, if necessary. This issue will be covered in Section 6.3.6.

6. Work Conserving Stateless Core Fair Queuing (C-SCORE)

6.1. Framework

FQ schedulers utilize the concept of FT that is used as the service order assigned to a packet. The packet with the minimum FT in a buffer is served first.

As an example, the VC scheduler [ZHANG] defines the FT to be

$$F(p) = \max\{F(p-1), A(p)\} + L(p)/r, \qquad (2)$$

where (p-1) and p are consecutive packets of the flow under observation, A(p) is the arrival time of p, L(p) is the length of p, and r is the flow service rate. The flow index is omitted.

The key idea of the FQ is to calculate the service completion times of packets in an imaginary ideal fluid service model and use them as the service order in the real packet-based scheduler.

While having the excellent flow isolation property, the FQ needs to maintain the flow state, F(p-1). For every arriving packet, the flow it belongs to has to be identified and its previous packet's FT should be extracted. As the packet departs, the flow state, F(p), has to be updated as well.

We consider a framework for constructing FTs for packets at core nodes without flow states. In a core node, the following conditions on FTs SHOULD be met.

C1) The 'fair distance' of consecutive packets of a flow generated at the entrance node has to be kept in the core nodes. That is; $Fh(p) >= Fh(p-1) + L(p)/r$, where Fh(p) is the F(p) at core node h.

C2) The order of FTs and the actual service order, within a flow, have to be kept. That is; $Fh(p) > Fh(p-1)$ and $Ch(p) > Ch(p-1)$, where Ch(p) is the actual service completion time of packet p at node h.

C3) The time lapse at each hop has to be reflected. That is; Fh(p) >= F(h-1)(p), where F(h-1)(p) is the FT of p at the node h-1, the upstream node of h.

In essence, (2) has to be approximated in core nodes. There can be many possible solutions to meet these conditions. We describe a generic framework with requirements for constructing FTs in core nodes, which are necessary to meet the conditions, without flow state, in the following.

Definition: An active period for a flow is a maximal interval of time during a node busy period, over which the FT of the most recently arrived packet of the flow is greater than the virtual time. Any other period is an inactive period for the flow.

Requirement 1: In the entrance node, it is REQUIRED to obtain the FTs with the following equation. 0 denotes the entrance node of the flow under observation.

$$F0(p) = \max\{F0(p-1), A0(p)\}+L(p)/r.$$

Note that if the FTs are constructed according to the above equation, the fair distance of consecutive packets is maintained.

Requirement 2: In a core node h, it is REQUIRED to increase the FT of a packet by an amount, d(h-1)(p), that depends on the previous node and the packet.

$$Fh(p) = F(h-1)(p) + d(h-1)(p).$$

Requirement 3: It is REQUIRED that dh(p) is a non-decreasing function of p, within a flow active period.

Requirements 1, 2, and 3 specify how to construct the FT in a network. By these requirements, Conditions C1), C2), and C3) are met. The following requirements 4 and 5 specify how the FT is used for scheduling.

Requirement 4: It is REQUIRED that a node provides service whenever there is a packet.

Requirement 5: It is REQUIRED that all packets waiting for service in a node are served in the ascending order of their FTs.

We call this framework the work conserving stateless core fair queuing (C-SCORE) [C-SCORE], which can be compared to the existing non-work conserving scheme [STOICA].

6.2. E2E Latency Bound

For C-SCORE to guarantee E2E latency bound, the dh(p) is RECOMMENDED to be defined as in the following.

$$dh(p) = SLh. \quad (3)$$

The service latency of the flow at node h, denoted by SLh, is given as follows.

$$SLh = Lh/Rh + L/r, \quad (4)$$

where Lh is the observed maximum packet length in the node h over all the flows, Rh is the link capacity of the node h, and L is the maximum packet length in the flow.

The concept of the service latency was first introduced in the Latency-rate server model [STILIADIS-LRS], which can be interpreted as the worst delay the first packet of a new flow can experience in the system.

Consider the worst case: Right before a new flow's first packet arrives at a node, the transmission of another packet with length Lh has just started. This packet takes the transmission delay of Lh/Rh. After the transmission of the packet with Lh, the flow under observation could take only the allocated share of the link, and the service of the packet under observation would be completed after L/r. Therefore, the packet has to wait, in the worst case, Lh/Rh + L/r.

The reason to add the service latency to F(h-1)(p) to get Fh(p) is to meet Condition C3) in a most conservative way without being too excessive. Intuitively, when every packet's FT is updated with the flow's own worst delay, then a packet that experienced the worst delay gets a favor. Thus its worst delay will not get any worse, while the delay differences among flows are reflected.

When dh(p) is decided by (3), then it can be proved that

$$Dh(p) <= (B-L)/r + SL0 + SL1 + ... + SLh, \quad (5)$$

where Dh(p) is the latency experienced by p from the arrival at the node 0 to the departure from node h; B, L, and r are the max burst of, max packet length of, and allocated rate to the flow under observation that p belongs to, respectively [KAUR].

Note that the latency bound in (5) is the same to the network where every node has a stateful FQ scheduler, including VC. The parameters in the latency bound are all intrinsic to the flow, except Lh/Rh.

6.3. Operational Procedure

6.3.1. Metadata

As a packet arrives at a core node, it carries metadata F(h-1)(p), d(h-1)(p), and L/r.

L/r should be kept in a packet in order to calculate dh(p) according to (3) and (4).

Fh(p) and dh(p) are dynamic and thus need to be updated at every hop.

Fh(p) can be obtained by a summation of the metadata F(h-1)(p) and d(h-1)(p).

dh(p) can be obtained by a summation of the metadata L/r and the node specific parameter Lh/Rh.

They can be updated during the time interval between the packet arrival to the switch (or router) and putting it into the queue in the output port.

Alternatively, Fh(p) can be pre-calculated at node h-1 with d(h-1)(p), which is the function of node h-1. In this case Fh(p) and L/r are the only metadata necessary. In Section 6.3.3 and Section 6.3.4, we follow this alternative approach.

The clear definition and format of metadata will be described in a later version.

6.3.2. Network Configuration for latency guarantee

A source requests E2E latency bound for a flow, specifying its arrival rate, maximum packet length, and maximum burst size. If the E2E latency bound can be met, the network admits the flow. In the process of admission decision, the service rate allocated to a flow can be decided according to the requested latency bound of the flow. The service rate should be chosen to be larger than or equal to the arrival rate of the flow. The network reserves the links in the path such that the sum of service rates to the flows does not exceed the link capacity.

The detailed operational procedure for such admission and reservation is out of scope of this document.

6.3.3. Role of entrance node for generation and update of FT

It is assumed that the packet length of p, L(p), is written in the packet header. Entrance node maintains the flow state, i.e. FT of packet (p-1) at node 0 (F0(p-1)), the maximum packet length of the flow (L), and the service rate allocated to the flow (r). It operates a clock to identify the arrival time of a packet (A0(p)). It collects the link information such as the maximum packet length of all the flow (L0) and link capacity (R0) to calculate the delay factor at the entrance node (d0(p)).

Upon receiving or generating packet p, it obtains F0(p) = max{F0(p-1), A0(p)} + L(p)/r, and uses it as the FT in the entrance node. If the queue is not empty then it puts p in a priority queue. It also obtains F1(p) = F0(p) + L0/R0 + L/r before or during p is in the queue. It writes F1(p) and L/r in the packet as metadata for use in the next node 1. Finally it updates the flow state information F0(p-1) to F0(p).

6.3.4. Role of core node for update of FT

A core node h collects the link information Lh/Rh. As in an entrance node, Lh is a rather static value, but still can be changed over time. Upon receiving packet p, it retrieves metadata Fh(p) and L/r, and uses Fh(p) as the FT value of the packet. It puts p in a priority queue. It obtains F(h+1)(p) = Fh(p) + Lh/Rh + L/r and updates the packet metadata Fh(p) with F(h+1)(p) before or during p is in the queue.

6.3.5. Mitigation of complexity in entrance node

Flow states still have to be maintained in entrance nodes. When the number of flows is large, maintaining flow states can be burdensome. However, this burden can be mitigated as follows. The notion of an entrance node can be understood as a various edge device, including a source itself. FT of a packet is decided based on the maximum of F0(p-1) and A0(p); and L(p)/r. These parameters are flow-specific. There is no need to know any other external parameters. The arrival time of p to the network, A0(p), can be defined as the generation time of p at the source. Then F0(p) is determined at the packet generation time and can be recorded in the packet. In other words, the entrance node functionality can reside in the source itself.

Therefore, we can significantly alleviate the complexity of the proposed framework. The framework is scalable and can be applied to any network.

6.3.6. Compensation of time difference between nodes

As it has been stated in Section 5, we have assumed zero propagation delays between nodes. In reality, there are time differences between nodes, including the differences due to the propagation delays. This time difference can be defined as the difference between the service completion time of a packet measured at the upstream node and the arrival time of the packet measured at the current node. In other words,

$$td(h-1, h)(p) = Ah(p) - C(h-1)(p),$$

where td(h-1, h)(p) is the time difference between node h-1 and h, and C(h-1)(p) is the service completion time measured at node h-1, for packet p respectively.

FT does not need to be precise. It is used just to indicate the packet service order. Therefore, if we can assume that the propagation delay is constant and the clocks do not drift, then td(h-1, h)(p) can be simplified to a constant value, td(h-1, h). In this case the delay factor in (3) can be modified to be

$$dh(p) = SLh + td(h, h+1).$$

The E2E latency bound in (5) increases as much as the sum of propagation delays from node 0 to h.

The time difference td(h-1, h) may be updated only once in a while.

By the time difference compensation, the nodes become aware of the global clock discrepancies using a periodic quantification of the local clock discrepancies between adjacent nodes. Link by link, this ends up producing awareness of the discrepancies between the clocks of all the nodes, which is then included in the computation of the FTs in core nodes. It is not synchronization in a strict sense because it does not involve the re-alignment of the clocks, only the quantification of their differences.

Even with the clock discrepancies and propagation delays, the framework in this document does not need global time synchronization.

6.4. Characteristics

The framework in this document, C-SCORE, is a flow level, rate based, work conserving, asynchronous, non-periodic, and in-time solution, according to the taxonomy suggested by [I-D.joung-detnet-taxonomy-dataplane].

Its per hop latency dominant factor is maximum packet length divided by service rate, independent of other flows' parameters. As such, its most distinguishable strength is the flow isolation capability. It can assign a fine tuned E2E latency bound to a flow, by controlling the flow's own parameters such as the service rate. Once the latency bound is assigned to the flow, then it remains almost the same in spite of the network situation changes, such as other flows' join and leave.

It is work conserving, thus enjoys the statistical multiplexing gain without wasting bandwidth, which is the key to the Internet's success. The consequence is a smaller average latency. The observable maximum latency is also much smaller than the theoretical latency bound. Note that, with a work conserving solution, observing the theoretical latency bound is extremely difficult. It is because the worst latency is an outcome of a combination of multiple rare events, e.g. a maximum burst from a flow collides with the maximum bursts from all other flows at every node. In contrast, non-work conserving solutions make it common to observe their latency bounds.

It is rate based, thus the admission condition check process is simple, which is dependent only on the service rates of flows. This process aligns well with existing protocols.

Overall, C-SCORE suits large scale networks, at any utilization level, with various types of flows join and leave dynamically.

7. IANA Considerations

There might be matters that require IANA considerations associated with metadata. If necessary, relevant text will be added in a later version.

8. Security Considerations

This section will be described later.

9. Acknowledgements

10. Contributor

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

[ANDREWS] Andrews, M., "Instability of FIFO in the permanent sessions model at arbitrarily small network loads", ACM Trans. Algorithms, vol. 5, no. 3, pp. 1–29, doi: 10.1145/1541885.1541894, July 2009.

[BOUILLARD] Bouillard, A., Boyer, M., and E. Le Corronc, "Deterministic network calculus: From theory to practical implementation", in Networks and Telecommunications. Hoboken, NJ, USA: Wiley, doi: 10.1002/9781119440284, 2018.

[C-SCORE] Joung, J., Kwon, J., Ryoo, J., and T. Cheung, "Scalable flow isolation with work conserving stateless core fair queuing for deterministic networking", IEEE Access, vol. 11, pp. 105225 – 105247, doi:10.1109/ACCESS.2023.3318479, 2023.

[GOLESTANI] Golestani, S. J., "A self-clocked fair queueing scheme for broadband applications", in Proc. INFOCOM, vol. 1, pp. 636–646, doi: 10.1109/INFCOM.1994.337677, 1994.

[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>.

[KAUR] Kaur, J. and H.M. Vin, "Core-stateless guaranteed rate scheduling algorithms", in Proc. INFOCOM, vol.3, pp. 1484–1492, 2001.

[PAREKH] Parekh, A. and R. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single-node case", IEEE/ACM Trans. Networking, vol. 1, no. 3, pp. 344–357, June 1993.

[STILIADIS-LRS]
Stiliadis, D. and A. Anujan, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms", IEEE/ACM Trans. Networking, vol. 6, no. 5, pp. 611-624, 1998.

[STILIADIS-RPS]
Stiliadis, D. and A. Anujan, "Rate-proportional servers: A design methodology for fair queueing algorithms", IEEE/ACM Trans. Networking, vol. 6, no. 2, pp. 164-174, 1998.

[STOICA] Stoica, I. and H. Zhang, "Providing guaranteed services without per flow management", ACM SIGCOMM Computer Communication Review, vol. 29, no. 4, pp. 81-94, 1999.

[THOMAS] Thomas, L., Le Boudec, J., and A. Mifdaoui, "On cyclic dependencies and regulators in time-sensitive networks", in Proc. IEEE Real-Time Syst. Symp. (RTSS), York, U.K., pp. 299-311, December 2019.

[ZHANG] Zhang, L., "Virtual clock: A new traffic control algorithm for packet switching networks", in Proc. ACM symposium on Communications architectures & protocols, pp. 19-29, 1990.

Authors' Addresses

Jinoo Joung
Sangmyung University
Email: jjoung@smu.ac.kr

Jeong-dong Ryoo
ETRI
Email: ryoo@etri.re.kr

Taesik Cheung
ETRI
Email: cts@etri.re.kr

Yizhou Li
Huawei
Email: liyizhou@huawei.com

Peng Liu
China Mobile

Email: liupengyjy@chinamobile.com

Network
Internet-Draft
Intended status: Standards Track
Expires: 10 October 2025

Shaofu. Peng
ZTE Corporation
Zongpeng. Du
China Mobile
Kashinath. Basu
Oxford Brookes University
Zuopin. Cheng
New H3C Technologies
Dong. Yang
Beijing Jiaotong University
Chang. Liu
China Unicom

8 April 2025

Deadline Based Deterministic Forwarding
draft-peng-detnet-deadline-based-forwarding-15

Abstract

This document describes a deadline based deterministic forwarding mechanism for IP/MPLS network with the corresponding resource reservation, admission control, scheduling and policing processes to provide guaranteed latency bound. It employs a latency compensation technique with a stateless core, to replace reshaping, making it suitable for the Differentiated Services (Diffserv) architecture [RFC2475].

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 10 October 2025.

Table of Contents

1. Introduction

[RFC8655] describes the architecture of deterministic network and defines the QoS goals of deterministic forwarding: Minimum and maximum end-to-end latency from source to destination, timely delivery, and bounded jitter (packet delay variation); packet loss ratio under various assumptions as to the operational states of the nodes and links; an upper bound on out-of-order packet delivery. In order to achieve these goals, deterministic networks use resource reservation, explicit routing, service protection and other means. Resource reservation provides dedicated resources (such as bandwidth, buffer space, time slots, etc.) to DetNet flows. Explicit routing ensures the stability of the route and does not change with the real-time change of network topology. Service protection reduces the packet loss by sending multiple DetNet flows along multiple disjoint paths at the same time.

[P802.1DC] described some Quality of Service (QoS) features specified in IEEE Std 802.1Q, such as per-stream filtering and policing, queuing, transmission selection, stream control and preemption, in a network system which is not a bridge. The internal structure of IP/MPLS routers may also be based on these components to describe the scheduling process of packets. In the presence of admission check, policing, reshaping, a large number of packet scheduling techniques can provide bounded latency. However, many solutions may result in an inefficient use of network resources, or provide an overestimated latency. Currently the underlying scheduling mechanisms in IP/MPLS networks generally use SP (Strict Priority) and WFQ (Weighted Fair Queuing), and manage a small number of priority based queues. They are rate based schedulers.

For SP, the highest priority queue can consume the total port bandwidth, while for WFQ scheduler, each queue may be configured with a pre-set rate limit. Both of them can provide the worst-case latency, but evaluation is generally overestimated. In the case where the network core supports reshaping per flow (or optimized reshaping as provided by [IR-Theory]), the worst-case latency of a flow is approximately equal to the aggregated burst of the traffic class divided by the rate limit of that traffic class. A rate-based scheduler may refer to [Net-Calculus] to obtain its rate-latency service curve and get a more tighter evaluation. When the network core does not implement reshaping, multiple flows sharing the same priority may form burst cascade, making it more difficult or even impossible to evaluate the worst-case latency of a single flow. [EF-FIFO] discusses the SP scheduling behavior in this core-stateless situation, which requires the overall network utilization level to be limited to a small portion of its link capacity in order to provide an appropriate bounded latency.

To address the overestimation issue of rate based scheduling, i.e., if want a low latency, may be forced to allocate a large service rate, according to [EDF-algorithm], an EDF (earliest-deadline-first) scheduler, which always selects the packet with the shortest deadline for transmission, is an optimal scheduler for a bounded delay service in the sense that it can support the delay bounds for any set of connections that can be supported by some other scheduling method. EDF is a delay-based scheduler, which further distinguishes flows in terms of time urgency, rather than rough traffic classes.

The academic community has conducted extensive research on EDF. [RPQ-EDF] proposed a method for implementing a rotating queue for EDF and its schedulability conditions. [Jitter-EDF] proposed a combination of damper and EDF to achieve low jitter. [RC-EDF] and [RC-EDF-para] proposed combining re-shaping per hop with EDF. [CQ-EDF] proposed a programmable calendar queues that enables the

efficient realization of EDF algorithm. [SCED] defined a deadline allocation algorithm that guarantees that a flow does have a minimum service curve.

This document introduces EDF scheduling mechanism to IP/MPLS network, as well as corresponding resource reservation, admission control, policing, etc, to provide guaranteed latency, as a supplement to IEEE 802.1 TSN mechanisms. It is a layer-3 solution and can operate over different types of QoS sensitive layer 2 including TSN but is not an alternative to TSN. Especially, a latency compensation based option is recommended to replace reshaping to be suitable for Diff-Serv architecture [RFC2475]. This document also discusses two scheduling behaviors: in-time scheduling and on-time scheduling. The former only provide bounded delay, while the latter further provide bounded jitter.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. EDF Scheduling Overview

The EDF scheduler assigns a deadline for each incoming packet, which is equal to the time the packet arrives at the node plus the latency limit, i.e., planned residence time (D), see Section 2.1. The EDF scheduling algorithm always selects the packet with the earliest deadline for transmission.

The precondition for EDF to work properly is that any DetNet flow must always satisfy the given traffic constraint function when it reaches a certain EDF scheduler. Therefore, it should generally implement traffic regulation at the network entrance to ensure that the admitted traffic complies with the constraints; And, implement reshaping on each intermediate node to temporarily cache packets to ensure that packets entering the EDF scheduler queue comply with the constraints. However, reshaping per flow is a challenge in large-scaling networks. Some core stateless optimization method need to be considered.

Another challenge of EDF scheduling is that queued packets must be sorted and stored according to their deadline, and whenever a new packet arrives at the scheduler, it needs to perform search and insert operations on the corresponding data structure, e.g., a PIFO (put-in first-out) queue, at line rate. Some approximate methods for sorted queues can be considered.

According to the above two challenges and the potential optimization methods, we will obtain four combination solutions. Operators should choose appropriate solutions based on the actual network situation. This document suggests using option-3 or option-4, which are referred to as CEDF (latency Compensation EDF). CEDF adjusts and sorts the arrival flows by the latency compensation factor carried in the packets, ensuring that the flows arrived at the EDF scheduler always conform to their constraints, avoiding the network core from maintaining the flow states, to meet large scaling requirements.

* option-1: Reshaping plus sorted queue.

* option-2: Reshaping plus RPQ.

* option-3: Latency Compensation plus sorted queue.

* option-4: Latency Compensation plus RPQ.

2.1. Planned Residence Time of the DetNet Flow

The planned residence time (termed as D) of the packet is an offset time, which is based on the arrival time of the packet and represents the maximum time allowed for the packet to stay inside the node.

For a deterministic path, the end-to-end delay includes two parts, the accumulated residence time and the accumulated link propagation delay. The accumulated residence time may be shared equally by each node along the path to obtain the average planned residence time of each node, or each node may have different planned residence time. The link propagation delay is generally constant, but not always so, for example, it may vary with temperature changes. Assuming that the tool for detecting the link propagation delay can sense the changes beyond the preset threshold and trigger the recalculation of the deterministic path.

The ingress PE node, when encapsulating DetNet flows, can explicitly insert the planned residence time into the packet according to SLA. The transit node, after receiving the packet, can directly obtain the planned residence time from the packet. Generally, only a single average planned residence time needs to be carried in the packet, which is shared to all nodes along the path; Or insert a stack of

planned residence time, one for each node.
[I-D.peng-6man-deadline-option] defined a method to carry the shared planned residence time in the IPv6 packets.
[I-D.pb-6man-deterministic-crh] and [I-D.p-6man-deterministic-eh] defined methods to carry the stack of planned residence time in the IPv6 packets.

An implementation should support the policy to forcibly override the planned residence time obtained from the packet.

2.2. Delay Levels Provided by the Network

The network may provide multiple delay levels on the outgoing port, each with its own delay resource pool. For example, some typical delay levels may be 10us, 20us, 30us, etc.

In theory, any additional delay level can be added dynamically, as long as the buffer and remaining bandwidth on the data plane allow.

The quantification of delay resource pool for each delay level is actually based on the schedulability conditions of EDF. This document introduces two types of resources per delay level:

* Burst: represents the amount of bits bound that a delay level provided.

* Bandwidth: represents the amount of bandwidth bound that a delay level provided. The bandwidth possessed by a certain delay level is also known as the service rate of that delay level.

For more information on the construction of resource pools, please refer to Section 3.2 and Section 4.3.

2.3. Relationship Between Planned Residence Time and Delay Level

The planned residence time (D) is the per-hop latency requirement of the flow, while the delay level (d) is the capability provided by the link.

Generally, we only need to design a limited number of delay levels to support a larger number of per-hop latency requirement. For example, there are delay levels such as d_1, d_2, ..., and d_n, In the resource management of the control plane, we assign d_i resources to all D that meet d_i <= D < d_(i+1).

2.4. Relationship Between Service Burst Interval and Delay Level

Although we generally prefer to have the service burst interval (SBI) greater than the maximum delay level, there is actually no necessary association between SBI and delay level.

Firstly, can a flow with small SBI (such as 10us) request a larger delay level (such as 100us)? Yes. It seems that during a longer residence time caused by delay level, there will be multiple rounds of burst interval packets leading to bursts accumulation. However, these packets can be distinguished and sent in sequence. In fact, we can multiply the original SBI by several times to obtain the expanded SBI (which includes multiple original bursts), with a length greater than the requested delay level, to get the preferred paradigm.

Secondly, can a flow with large SBI (such as 1ms) request a smaller delay level (such as 10us)? This is certainly yes.

3. Sorted Queue

[PIFO] defined the push-in first-out queue (PIFO), which is a priority queue that maintains the scheduling order or time. A PIFO allows elements to be pushed into an arbitrary position based on an element's rank (the scheduling order or time), but always dequeues elements from the head.

3.1. Scheduling Mode for PIFO

A PIFO queue may be configured as either in-time or on-time scheduling mode, but cannot support both modes simultaneously.

In the in-time scheduling mode, as long as the queue is not empty, packets always depart from the head of queue (HoQ) for transmission. The actual bandwidth consumed by the scheduler may exceed its preset service rate C.

In the on-time scheduling mode, if the queue is not empty and the rank of the HoQ packet is equal to or earlier than the current system time, then the HoQ packet will be sent. Otherwise, not.

3.2. Schedulability Condition for PIFO

[RPQ-EDF] has given the schedulability condition for classic EDF that is based on any type of sorted queue with in-time scheduling mode.

Suppose for any delay level d_i, the corresponding accumulated constraint function is A_i(t). Let d_i < d_(i+1), then the schedulability condition is:

sum{A_i(t-d_i) for all i} <= C*t (Equation-1)

where, C is service rate of the EDF scheduler.

It should be noted that for a delay level d_i, its residence time is actually contributed by its own flows and all other more urgent delay levels. Based on the schedulability conditions, we can choose the traffic arrival constraint function according to the preset delay level, or we can choose the delay level according to the preset traffic arrival constraint function.

When setting up new flows in the network, admission check based on schedulability condition must be executed on each link that the flow passes through.

Here, A_i(t) defines the upper limit of eligible arrivals of delay level d_i, and should not be treated as the actual arrivals (we mark it as a_i(t) for distinction). As described in this document, a_i(t) may contain ineligible arrivals that need first to be converted (or sorted) into eligible arrivals, e.g., by method of regulation (Section 5) or latency compensation (Section 6), and then processed by the EDF scheduler.

3.2.1. Schedulability Conditions for Leaky Bucket Constraint Function

Assume that we want to support n delay levels (d_1, d_2,..., d_n) in the network, and the traffic arrival constraint function of each delay level d_i is the leaky bucket arrival curve A_i(t) = b_i + r_i * t. Equation-1 can be expressed as:

b_1 <= C*d_1 - M

b_1 + b_2 + r_1*(d_2-d_1) <= C*d_2 - M

b_1 + b_2 + b_3 + r_1*(d_3-d_1) + r_2*(d_3-d_2) <= C*d_3 - M

... ...

sum(b_1+...+b_n) + r_1*(d_n-d_1) + r_2*(d_n-d_2) + ... + r_n_1*(d_n-d_n_1) <= C*d_n - M

where, C is the service rate of the EDF scheduler, M is the maximum size of the interference packet.

Note that the preset value of b_i does not depend on r_i, but r_i generally refers to b_i (and burst interval) for setting. For example, the preset value of r_i may be small, while the value of b_i may be large. Such parameter design is more suitable for transmitting traffic with large service burst interval, and large service burst size, but small bandwidth requirements.

An extreme example is that the preset r_i of each level d_i is close to 0 (this is because the burst interval of the served flow is too large, e.g., one hour or one day), but the preset b_i is close to the maximum value (e.g., b_1 = C*d_1 - M), then when the concurrent flow of all delay levels is scheduled, the time 0~d_1 is all used to send the burst b_1, the time d_1~d_2 is all used to send the burst b_2, the time d_2~d_3 is all used to send the burst b_3, and so on.

However, the typical allocation scheme is that the preset r_i of each level d_i will divide C roughly equally. For example, we may firstly pre-allocate b_1 = C*d_1 - M, r_1 = C/n; Then recursively pre-allocate b_2 = C*(d_2-d_1)*(n-1)/n, r_2 = C/n; And so on. The pre-allocated parameters b_i and r_i of each level d_i constitute the delay resources of that level of the link. A path can reserve required burst and bandwidth from delay resources of the specific delay level d_i, and the reservation is successful only if the two resources are successfully reserved at the same time. As long as neither b_i nor r_i is free, the delay resource of level d_i is exhausted.

Alternatively, a more tight allocation scheme is to not preset the parameters of A_i(t), but to dynamically accumulate the parameters of A_i(t) based on the actual flows setup demand, and always check whether the schedulability condition is met based on the updated A_i(t) during the flow setup procedure. In this case, it is still necessary to set a resource limit for each delay level to prevent the flows of a certain delay level from consuming all resources. For example, we may set the resource limit of each delay level d_i to b_i_limit = C*(d_i - d_(i-1)) - M, r_i_limit = C/n. In this case, the dynamically updated b_i and r_i should be treated as utilized resources, and participate in schedulability condition checks.

Note that for some delay level d_i, its resource may be explicitly set to empty, i.e., b_i = 0, r_i = 0. This brings flexibility, and resources can be freed up for later delay levels with lower priority to use.

In a specific scenario, if the ideal arrival packet interval (by the method of re-shaping or latency compensation) of all service flows is large, and the maximum delay level d_n chosen is not larger than any packet interval of any service flow, the schedulability condition can be further simplified as follows:

```
b_1 <= C*d_1 - M, r_1 = b_1 / d_n;

b_1 + b_2 <= C*d_2 - M, , r_2 = b_2 / d_n;

b_1 + b_2 + b_3 <= C*d_3 - M, r_3 = b_3 / d_n;

... ...

sum(b_1+...+b_n) <= C*d_n - M, r_n = b_n / d_n;
```

The above simplified condition implies that the total number of bursts contained within any time interval d_n does not exceed sum(b_1+...+b_n). This is true because for any flow i it never contains two packets in a single time interval d_n. In this case, it can support a larger service scale than the original condition.

It should be noted that the burst and bandwidth resource of each delay level mentioned above always assumes that the flows it serves arrive concurrently from many incoming interfaces (i.e., with a large concurrency), which is a safe but conservative assumption. If operators are aware of the specific topology knowledge of the network, such as having very little (or even no) concurrency, they can design special resource pools.

For example, in the case of one incoming one outgoing, there will be no queueing delay, and a single delay level can be used for all interleaved flows. In this case, the delay level value just equals the forwarding delay (F), plus the transmission delay of a single packet. There is no limit on burst resources, and the upper limit of bandwidth resources is still the service rate C. Alternatively, a simple FIFO queueing mechanism can also work in this case.

For example, in the case of multiple incoming one outgoing, if the burst size of each incoming interface is known, the resolved size can be calculated (i.e., the sum of all non longest burst size) and used to design a single delay level. In this case, the delay level value equals the forwarding delay (F), plus the transmission delay of the resolved size. The burst resources should not exceed the sending volume corresponding to the burst interval, and the upper limit of bandwidth resources is still the service rate C. However, it is also possible to design more delay levels, each for a different subset of flows. In this case, the burst resource of urgent delay level must also be limited to avoid large value for other delay levels.

3.2.2. Schedulability Condition Analysis for On-time Mode

Compared with in-time mode, on-time mode is non-work-conserving, which can be considered as the combination of damper and EDF scheduler. On-time scheduling mode applied on a flow try to maintain the packet interval between any adjacent packets of that flow to be consistent with the regulated interval on the flow entrance node. The maintenance of packet intervals does not lead to an increase in the bandwidth occupied by that flow and cause the arrival curve to violate the traffic constraint function. So that the schedulability condition (i.e., Equation-1) can also be applied to on-time scheduling mode. See Section 8 for more information about jitter control.

3.3. Buffer Size Design

The service rate of the EDF scheduler, termed as C, can reach the link rate, but generally only needs to be configured as part of the link bandwidth, such as 50%. Allow hierarchical scheduling, for example, the EDF scheduler may participate in higher-level WFQ scheduling along with other schedulers.

If flows are rate-controlled (i.e., reshaping is done inside the network, or on-time mode is applied), the maximum depth of PIFO should be C * d_n, where d_n is the maximum delay level. Otherwise, more buffer is necessary to absorb the burst accumulation. The PIFO zone where the distance from HoQ exceeds the maximum delay level is just used to store accumulated bursts. Please refer to Section 12 for more considerations.

4. Rotation Priority Queues

[RPQ-EDF] described rotating priority queues, and the priority granularity of the queue is the same as the rotation interval. However, if the planned residencetime of the flow is used as priority, it requires a lot of priority and corresponding queues. Therefore, this section provides a limited nunmber of rotating priority queues with count-down time range whose rotation interval is more refined, with the following characteristics:

* Each queue has CT (Count-down Time) that is decreased by RTI (Rotation Time Interval). The CT difference between two adjacent queues is CTI (CT Interval). RTI must be less than or equal to CTI, with CTI = K * RTI, where the natural number K >= 1.

* The smaller the CT, the higher the priority. At the beginning, all queues have different initial CT values, i.e., staggered from each other, e.g., one queue has the minimum CT value (termed as MIN_CT), and one queue has the maximum CT value (termed as MAX_CT), and the CT values of all queues increase equally by CTI. Note that CT is just the countdown of the HoQ, and the countdown of the end of the queue (EoQ) is near CT+CTI. So the CT attribute of a queue is actually a range [CT, CT+CTI).

* For a queue whose CT is MIN_CT, after a new round of CTI, its CT will become MIN_CT - CTI and immediately return to MAX_CT.

The above CTI, RTI, MIN_CT and MAX_CT value should be chosen according to the hardware capacity. Each link can independently use different CTI. The general principle is that the larger bandwidth, the smaller CTI. The CTI must be designed large enough to include interference delay caused by a single packet with maximum size.

The choose of RTI should consider the latency granularity of various DetNet flows, so that CT updated per RTI can match the delay requirements of different flows. One implementation may not choose to let CT be actually updated at the granularity of RTI, but at the granularity of CTI. For example, the elapsed time within CTI can be recorded, and (cur_CT - elapsed_time) can be used as the actual CT of the queue, where cur_CT is the current CT of the queue that has not been updated yet. Although the update of cur_CT is slow, the actual CT is sensitive enough.

According to different scheduling mode configured to the RPQ, MIN_CT may be designed to different values. For in-time mode, MIN_CT may be 0. For on-time mode with option E|D decoupling (see Section 8), MIN_CT may also be 0. For on-time mode with option E+D integration, MIN_CT may be -N*CTI, where N is the amount of delay levels.

A specific example of RPQ configured with in-time scheduling mode is depicted in Figure 1.

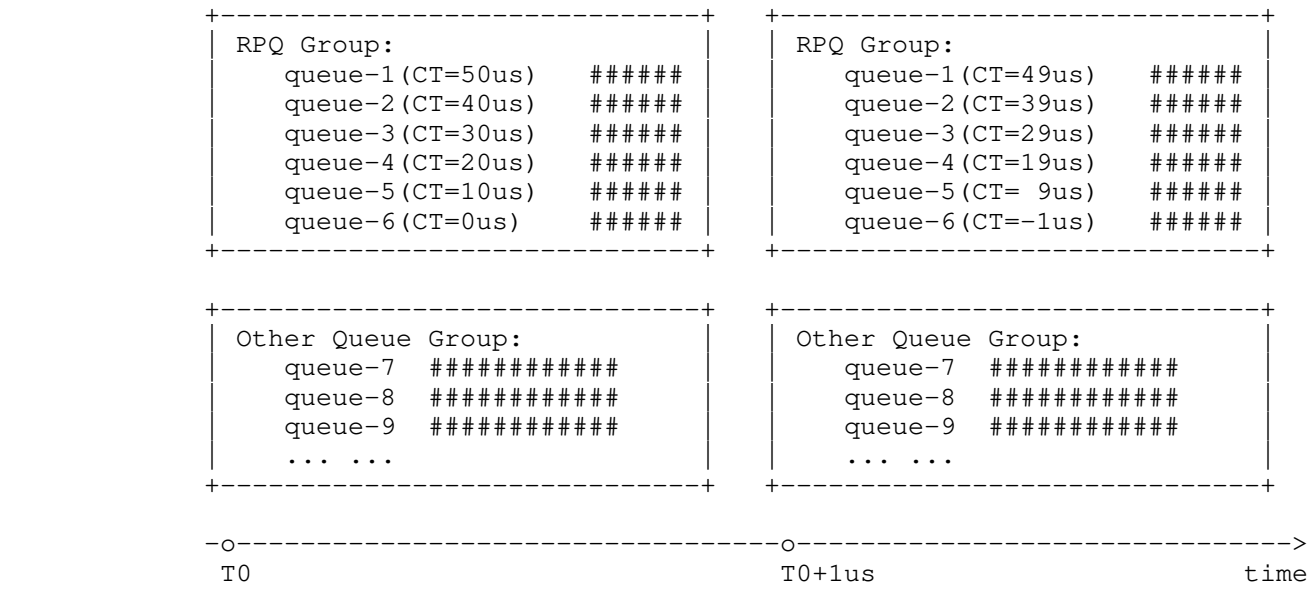

Figure 1: Example of RPQ Groups

In this example, the CTI for RPQ group is configured to 10us. Queue-1 ~ queue-6 are members of RPQ group. Each queue has its initial CT attribute, and the CT of all queues are staggered from each other. For example, the CT of queue-1 is 50us (MAX_CT), the CT of queue-2 is 40uS, ..., the CT of queue-6 is 0 (MIN_CT).

Suppose the scheduling engine initiates a rotation timer with a time interval of 1us, i.e., CTI = 10 * RTI in this case. As shown in the figure, at T0 + 1us, the CT of queue-1 becomes 49us, the CT of queue-2 becomes 39us, etc.

At T0 + 10us, the CT of queue-6 will return to 50us (MAX_CT).

Note that the minimum D requested by a DetNet flow should not be smaller than d_1+F, where d_1 is the most urgent delay level, F is the intra node forwarding delay. Therefore any packets with in-time scheduling should have Q (i.e., D + E - F) that is not be smaller than d_1, and should never be inserted to a queue whose CT is negative.

4.1. Alternate Queue Allocation Rules

There may be extreme scenarios that multiple delay levels of eligible bursts arrive sequentially, with lower priority burst arriving first and higher priority burst arriving later, and then simultaneously releasing flood. In this case, it is necessary to ensure that the higher priority burst is sent first to meet its deadline.

Therefore it may further let a RPQ queue (act as the virutal parent queue) contain multiple sub-queues, each for a delay level. The physical sub-queue with small delay level (e.g., 10us) is ranked before the physical sub-queue with large delay level (e.g., 20us). Packets are actually stored in the physical sub-queues. That is, packets belonging to different delay levels are inserted into different sub-queues and protected. In this way, for two packets with the same Q but different D, we can decide to firstly schedule the packet with the smallest D.

This alternate queue allocation rule enables eligible arrivals always have a place to store, avoiding conflicts in local positions of the RPQ queue group and causing overflow.

4.2. Scheduling Mode for RPQ

A RPQ group may be configured as either in-time or on-time scheduling mode, but cannot support both modes simultaneously.

In the in-time scheduling mode, in all non empty queues, the packets in each queue are sequentially sent in the order of high priority queue to low priority queue. The actual bandwidth consumed by the scheduler may exceed its set service rate C.

In the on-time scheduling mode, only in all non empty queues with CT <= 0, the packets in each queue are sequentially sent in the order of high priority queue to low priority queue.

For a virtual parent queue that is allowed to be sent, for the multiple non empty physical sub-queues it contains, packets are sequentially sent from the non empty physical sub-queues along the direction from the physical sub-queues with small delay levels to the physical sub-queues with large delay levels. Only when a physical sub-queue is cleared can the next non empty physical sub-queue be sent.

4.3. Schedulability Condition for RPQ

In this section, we discuss the schedulability condition based on RPQ with in-time scheduling mode firstly.

Suppose for any delay level d_i, the corresponding accumulated constraint function is A_i(t), and let d_i < d_(i+1). Suppose for any planned residence time D_i, the the corresponding constraint function is A'_i(t). For simplicity, we take intra node forwarding delay F as 0. Then the schedulability condition is:

* A_1(t-d_1) + sum{A_i(t+CTI-d_i) for all i>=2} <= C*t, if a d_i contains only one D_i. (Equation-2)

* sum{A_i(t+CTI-d_i) for all i>=1} <= C*t, if d_i contains multiple D_i. (Equation-3)

where CTI is the CT interval between adjacency queue, C is service rate of the EDF scheduler.

The proof is similar with that in [RPQ-EDF], except that the rotation interval is fine-grained by RTI and the priority of each queue is CT range. Please refer to Appendix A.

Note that the key difference between the above two conditions (i.e., Equation-2, Equation-3) and one based on sorted queue (i.e., Equation-1) is the CTI factor.

Other common considerations are the same as Section 3.2.

4.3.1. Schedulability Condition for Alternate QAR

According to Section 4.1, a RPQ queue may further contain multiple sub-queues, each for a delay level. Under the same parent queue, all sub-queues are sorted in descending order of delay level. In this case, the precise workload should exclude packets with higher delay levels than the observed packet.

In the case that d_i contains only one D_i, the schedulability condition is Equation-1.

In the case that d_i contains multiple D_i, the schedulability condition is still Equation-3.

Please refer to Appendix B.

4.3.2. Schedulability Conditions for Leaky Bucket Constraint Function

Assume that we want to support delay levels (d_1, d_2,..., d_n) in the network, and the traffic arrival constraint function of each delay level d_i is the leaky bucket arrival curve A_i(t) = b_i + r_i * t. Equation-2 can be expressed as:

```
b_1 <= C*d_1 - M

b_1 + b_2 + (r_1+r_2)*CTI <= C*d_2 - M

b_1 + b_2 + b_3 + (r_1+r_2)*2*CTI + r_3*CTI <= C*d_3 - M

... ...

sum(b_1+...+b_n) + (r_1+r_2)*(n-1)*CTI + r_3*(n-2)*CTI + ... +
r_n*CTI <= C*d_n - M
```

where, C is the service rate of the EDF scheduler, M is the maximum size of the interference packet.

Equation-3 can be expressed as:

```
b_1 + r_1*CTI <= C*d_1 - M

b_1 + b_2 + r_1*2*CTI + r_2*CTI <= C*d_2 - M

b_1 + b_2 + b_3 + r_1*3*CTI + r_2*2*CTI + r_3*CTI <= C*d_3 - M

... ...

sum(b_1+...+b_n) + r_1*n*CTI + r_2*(n-1)*CTI + ... + r_n*CTI <=
C*d_n - M
```

Similarly, in a specific scenario, if the ideal arrival packet interval (by the method of re-shaping or latency compensation) of all service flows is large, and the maximum delay level d_n chosen is not larger than any packet interval of any service flow, the above two schedulability conditions can be further simplified as follows:

```
b_1 <= C*d_1 - M, r_1 = b_1 / d_n;

b_1 + b_2 <= C*d_2 - M, , r_2 = b_2 / d_n;

b_1 + b_2 + b_3 <= C*d_3 - M, r_3 = b_3 / d_n;

... ...

sum(b_1+...+b_n) <= C*d_n - M, r_n = b_n / d_n;
```

4.3.3. Schedulability Condition Analysis for On-time Mode

Compared with in-time mode, on-time mode is non-work-conserving, which can be considered as the combination of damper and EDF scheduler. On-time scheduling mode applied on a flow try to maintain the packet interval between any adjacent packets of that flow to be consistent with the regulated interval on the flow entrance node. The maintenance of packet intervals does not lead to an increase in the bandwidth occupied by that flow and cause the arrival curve to violate the traffic constraint function. So that the schedulability condition (i.e., Equation-2/3) can also be applied to on-time scheduling mode. See Section 8 for more information about jitter control.

4.4. Buffer Size Design

An implementation may let all queues share the common buffer. Especially if Alternet QAR (Section 4.1) is applied, the actual buffer cost of a virtual parent queue is contributed by all the physical sub-queues it contains. The actual buffer cost of each physical sub queue is dynamically allocated based on whether there is a packet inserted. According to Section 4.3, the maximum buffer cost of a physical sub-queue may reach the upper limit of burst resources for the corresponding delay level.

If flows are rate-controlled (i.e., reshaping is done inside the network, or on-time scheduling mode is applied), the MAX_CT may be designed as the maximum delay level, and total necessary buffer shared by all queues should be C * d_n, where C is the service rate and d_n is the maximum delay level. Otherwise, MAX_CT should be larger than the maximum delay level, and with more necessary buffer, to absorb the burst accumulation. All the queues with CT larger than the maximum delay level are just used to store accumulated bursts. Please refer to Section 12 for more considerations.

5. Reshaping

Reshaping per flow inside the network, as described in [RFC2212], is done at all heterogeneous source branch points and at all source merge points, to restore (possibly distorted) traffic’s shape to conform to the TSpec. Reshaping entails delaying packets until they are within conformance of the TSpec.

A network element MUST provide the necessary buffers to ensure that conforming traffic is not lost at the reshaper. Note that while the large buffer makes it appear that reshapers add considerable delay, this is not the case. Given a valid TSpec that accurately describes the traffic, reshaping will cause little extra actual delay at the reshaping point (and will not affect the delay bound at all).

Maintaining a dedicated shaping queue per flow can avoid burstiness cascading between different flows with the same traffic class, but this approach goes against the design goal of packet multiplexing networks. [IR-Theory] describes a more concise approach by maintaining a small number of interleaved regulators (per traffic class and incoming port), but still maintaining the state of each flow. With this regulator, packets of multiple flows are processed in one FIFO queue and only the packet at the head of the queue is examined against the regulation constraints of its flow. However, as the number of flows increases, the IR operation may become burdensome as much as the per-flow reshaping.

For any observed EDF scheduler in the network, when the traffic arriving from all incoming ports is always reshaped, then these flows comply with their arrival constraint functions.

6. Latency Compensation

[RFC9320] presents a latency model for DetNet nodes. There are six type of delays that a packet can experience from hop to hop. The processing delay (type-4), the regulator delay (type-5) , the queueing subsystem delay (type-6), and the output delay (type-1) together contribute to the residence time in the node.

In this document, the residence time in the node is simplified into two parts: the first part is to lookup the forwarding table when the packet is received from the incoming port (or generated by the control plane) and deliver the packet to the line card where the outgoing port is located; the second part is to store the packet in the queue of the outgoing port for transmission. These two parts contribute to the actual residence time of the packet in the node. The former can be called forwarding delay (termed as F) and the latter can be called queueing delay (termed as Q). The forwarding delay is related to the chip implementation and is generally constant (with a maximum value); The queueing delay is unstable.

6.1. Accumulated Residence Time Deviation

The accumulated residence time deviation, also termed as latency deviation (E), equals accumulated planned residence time minus accumulated actual residence time. This value can be zero, positive, or negative.

The accumulated planned residence time of the packet refers to the sum of the planned residence time of all upstream nodes before the packet is transmitted to the current node. The accumulated actual residence time of the packet, refers to the sum of the actual residence time of all upstream nodes before the packet is transmitted to the current node.

In the case of in-time scheduling, E may be a very large positive value. While in the case of on-time scheduling, E may be 0, or a small value close to 0.

The setting of the latency deviation (E) of the packet needs to be friendly to the chip for reading and writing. For example, it should be designed as a fixed position in the packet. The chip may support flexible configuration for that position.

[I-D.peng-6man-delay-options] defined the method for carrying the latency deviation (E) in the IPv6 Hop-by-Hop Options Header. [I-D.pb-6man-deterministic-crh], [I-D.p-6man-deterministic-eh] defined methods for carrying the latency deviation (E) in the IPv6 Routing Header.

6.2. Allowable Queueing Delay

When an EDF scheduler receives a packet, it can calculate allowable queueing delay (Q) for the packet. Specifically, it can first get the latency deviation (E), and add it to the planned residence time (D) of the packet at this node to obtain the adjustment residence time, and then deduct the actual forwarding delay (F) of the packet in the node.

* Q = D + E - F

The scheduler selects a buffer position (e.g., queue-id, or rank) for the packet based on Q.

Note that one implementation may calculate Q at incoming port and determine the buffer position of the outgoing port. In this case, Q = D + E, and a buffer position indication may be notified from the incoming port to the outgoing port.

Assuming that the current node in a deterministic path is h, all upstream nodes are from 1 to h-1. For any node h, denote the planned residence time as D[h], the actual residence time as R[h], the input latency deviation (contributed by all upstream nodes) as E[h], the forwarding delay intra-node as F[h], then the allowable queueing delay (Q) of the packet on node h, i.e., Q[h], is:

$$Q[h] = D[h] + E[h] - F[h]$$

$$E[h] = D[h-1] + E[h-1] - R[h-1]$$

$$D[0], E[0], R[0] = 0$$

6.3. Scheduled by Allowable Queueing Delay

The packet will be scheduled based on its Q that is affected by latency compensation. The earliest literature similar to the idea of latency compensation based on E can be found in [Jitter-EDF].

The core stateless latency compensation can achieve the effect of reshaping per flow to get the eligible arrivals pattern. Q can be used to sort ineligible arrivals of one delay level and prevent them from interfering with the scheduling of eligible arrivals of other delay levels.

Firstly, at the flow (e.g., flow i) entrance node, all packets (after regulation) of flow i will be released to the EDF scheduler one after another at different time (termed as ideal arrival time), but with the same allowable queueing delay (Q), with initial E = 0, i.e., Q = D, assuming no link propagation delay and intra-node forwarding delay for simplicity. We denote this arrival pattern faced by the scheduler on the flow entrance node as arrival_pattern_0, which contains a sequence of packets with variant of intervals between adjacent packets. We say that arrival_pattern_0 is eligible arrivals because its arrival curve is less than the constraint function $A_i(t)$. For any packet p in arrival_pattern_0, assuming its ideal arrival time is t_p_0.

Then, we can get arrival_pattern_1 = arrival_pattern_0 + D that is also eligible arrivals, where, arrival_pattern_0 + D means that the ideal arrival time (at the scheduler of flow entrance node) of each packet in arrival_pattern_0 is added with D. In fact, arrival_pattern_1 is the eligible arrivals on the second node. That is, the second node may recover the eligible arrivals arrival_pattern_1 from the actual arrivals with the help of latency compensation, and then to schedule based on arrival_pattern_1. How did arrival_pattern_1 recover? For any packet p, assuming it experiences an actual queuing delay q on the flow entrance node, and

will actually arrive at the second node at time t_p_0 + q, with E = D - q carried in the sending packet. The second node will recover the eligible arrival time of packet p by, eligible arrival time = actual arrival time + E = t_p_0 + q + D - q = t_p_0 + D. Therefore arrival_pattern_1 is recovered.

Similarly, the third node may recover arrival_pattern_2 = arrival_pattern_0 + 2*D, and the fourth node may recover arrival_pattern_3 = arrival_pattern_0 + 3*D, and so on. On any node h, packet p will be sorted in the scheduler queue based on its ideal departure time (i.e., eligible arrival time plus D) for scheduling.

Because the scheduler always schedules based on eligible arrivals, its scheduling power will not be overwhelmed by actual arrivals that may include burst accumulation.

We may think the packets sorted in the queue with ideal departure time as a virtual regulation, because the rank distance between the adjacent packets of the flow i is exactly maintained consistently with the corresponding regulated interval between these two adjacent packets on the flow entrance node. There is no need to require that this virtual regulation and a real regulation component must have exactly the same pattern, as long as each pattern is less than the arrival constraint function A_i(t).

Although, latency compensation has the effect of reshaping, but it is not equivalent to reshaping. Considering an accumulated bursts that violates the traffic constraint function and arrives at a node, if reshaping is used, it will substantially introduce shaping delay for the ineligible bursts, which will then enter the queueing subsystem. While if latency compensation is used, this ineligible bursts will only be penalized with a larger Q and tolerated to be placed in the queueing sub-system, and in the case of in-time mode it may be immediately sent if higher priority queues are empty.

Note that the premise of latency compensation is that a flow must be based on a fixed explicit path. If multiple packets from the same flow arrive at the intermediate node via multiple paths with different propagation lengths, even if these packets are all eligible, bursts accumulation may still form and cannot even be punished.

7. Solution Options

7.1. Option-1: Reshaping plus Sorted Queue

As shown in Figure 2, a received packet is inserted to the PIFO queue according to rank = A + D - F, where, A is the time that packet arrived at the scheduler, i.e., arrive_time_S in the figure. Depending on the situation of the accumulated burst arrived at the input port, different packets may face different shaping delays. The shaper will convert the input ineligible arrivals pattern (if possible) into an eligible arrivals pattern. Here, D - F may be denoted as the allowable queueing delay Q.

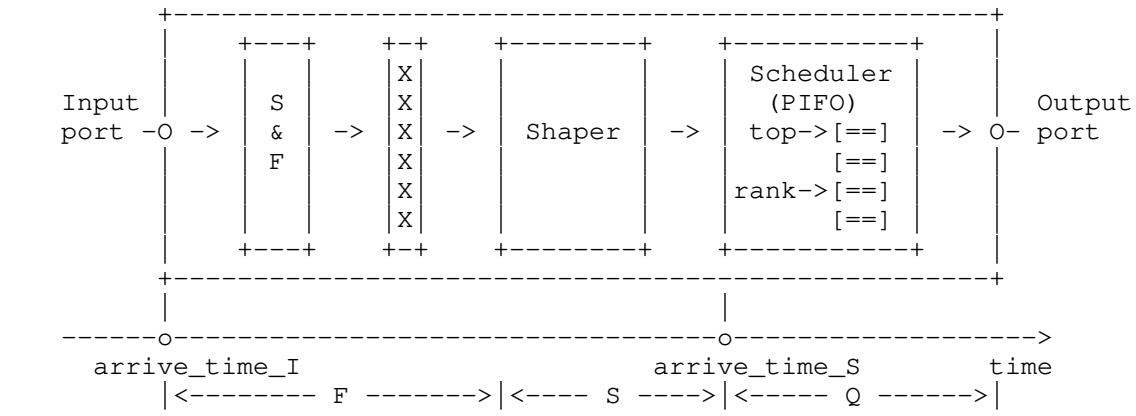

Figure 2: Reshaping plus Sorted Queue

Enqueue rule:

* For two packets with different rank, the packet with a smaller rank is closer to the head of the queue.

* For two packets with the same rank, the packet with a smaller D is closer to the head of the queue.

* For two packets with the same rank and D, the packet that arrive at the scheduler first is closer to the head of the queue.

The planned residence time (D) should be carried in the packet.

The scheduling mode (in-time or on-time) should also be carried in the packet, and used to insert packet into PIFO with the corresponding scheduling mode.

Dequeue rule:

* As mentioned in Section 3.1, for a PIFO with in-time scheduling mode, as long as the queue is not empty, packets always departure from the HoQ for transmission; while for PIFO with on-time scheduling mode, only if the queue is not empty and the rank of the HoQ packet is equal to or earlier than the current system time, the HoQ packet can be sent.

However, in this option the dequeue rule of on-time mode can not guarantee jitter, due to lack of factor E to absorb jitter per hop. The dequeue rule of on-time mode only controls the starting time when packets are allowed to be scheduled, but cannot guarantee that different packets have the same queuing delay.

7.2. Option-2: Reshaping plus RPQ

As shown in Figure 3, a received packet is inserted to the appropriate RPQ queue with specific CT to meet CT <= Q < CT+CTI when the packet arrived at the scheduler, where Q = D - F. Depending on the situation of the accumulated burst arrived at the input port, different packets may face different shaping delays. The shaper will convert the input ineligible arrivals pattern (if possible) into an eligible arrivals pattern.

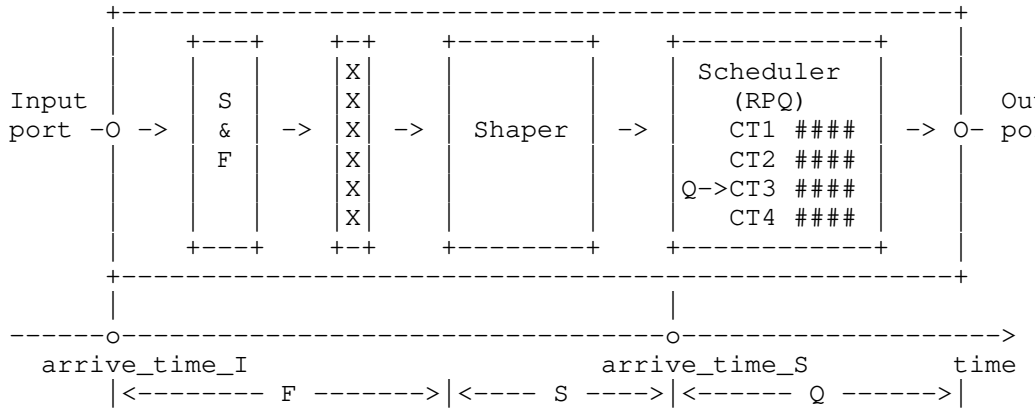

Figure 3: Reshaping plus RPQ

Enqueue rule:

* For a packet with Q, select the target RPQ queue (i.e., the virtual parent queue) with corresponding CT, that meet CT <= Q < CT+CTI.

* Under the selected virtual parent queue, select the target physical sub-queue with corresponding delay level d_i, which is closest to D-F and not greater than D-F.

The planned residence time (D) should be carried in the packet.

The scheduling mode (in-time or on-time) should also be carried in the packet, and used to insert packet into RPQ with the corresponding scheduling mode.

Dequeue rule:

* As mentioned in Section 4.2, for a RPQ group with in-time scheduling mode, in all non empty queues, the packets in each queue are sequentially sent in the order of high priority queue to low priority queue; while for a RPQ group with on-time scheduling mode, only in all non empty queues with CT <= 0, the packets in each queue are sequentially sent in the order of high priority queue to low priority queue.

However, in this option the dequeue rule of on-time mode can not guarantee jitter, due to lack of factor E to absorb jitter per hop. The dequeue rule of on-time mode only controls the starting time when packets are allowed to be scheduled, but cannot guarantee that different packets have the same queuing delay.

7.3. Option-3: Latency Compensation plus Sorted Queue

As shown in Figure 4, a received packet is inserted to the PIFO queue according to rank = A1 + E + D, or rank = A2 + E + D - F, where, A1 is the time that packet arrived at the input port (i.e., arrive_time_I in the figure), A2 is the time that packet arrived at the scheduler (i.e., arrive_time_S in the figure). Note that E is initially 0 on the flow entrance node, and generally not 0 on other nodes and will update per hop. Depending on the situation of the accumulated burst arrived at the input port, different packets may have different input latency deviation E. Latency compensation will convert the input ineligible arrivals pattern (if possible) into an eligible arrivals pattern. Here, E + D - F may be denoted as the allowable queueing delay Q.

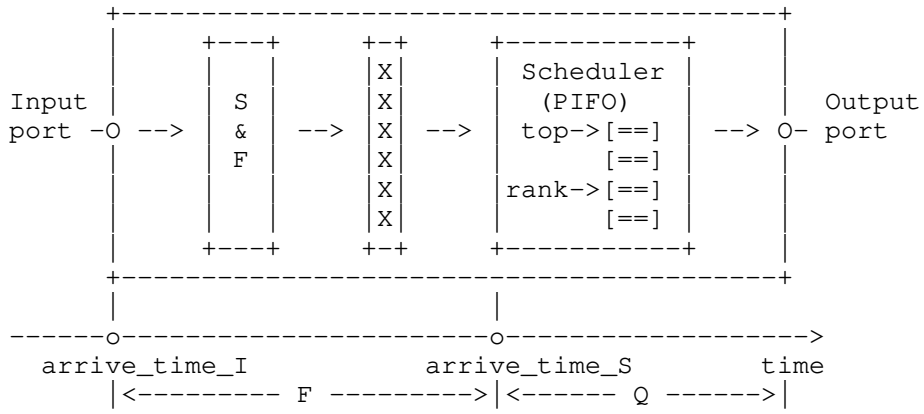

Figure 4: Latency Compensation plus Sorted Queue

The planned residence time (D) and latency deviation (E) should be carried in the packet.

The enqueue and dequeue operations are the same as Section 7.1.

In this option the dequeue rule of on-time mode can guarantee jitter with the help of factor E to absorb jitter per hop. See Section 8 for more information.

7.3.1. Packet Disorder Considerations

Suppose that two packets, P1, P2, are generated instantaneously from a specific flow at the source, and the two packets have the same planned residence time. P1 may face less interference delay than P2 in their journey. When they arrive at an intermediate node in turn, P2 will have less allowable queueing delay (Q) than P1 to try to stay close to P1 again. It should be noted that to compare who is earlier is based on the time arriving at the scheduler plus packet's Q. The time difference between the arrival of two packets at the scheduler may not be consistent with the difference between their Q. It is possible to get an unexpected comparison result.

As shown in Figure 5, P1 and P2 are two back-to-back packets belonging to the same flow. The arrival time when they are received on the scheduler is shown in the figure. Suppose that the Q values of two adjacent packets P1 and P2 are 40us and 39us, and arrive at the scheduler at time T1 and T2 respectively. P1 will be sorted based on T1 + 40us, while P2 will be sorted based on T2 + 39us. Ideally, T2 should be T1 + 1us. However, this may be not the case. For example, it is possible that T2 = T1 + 0.9us, Q1 = 40, Q2 = 39.1, but just because the calculation accuracy of Q1 and Q2 is

microseconds, so they are, e.g., with half-adjust, approximately 40 us and 39 us, respectively. This means that P2 will be sorted before P1 in the PIFO, resulting in disorder.

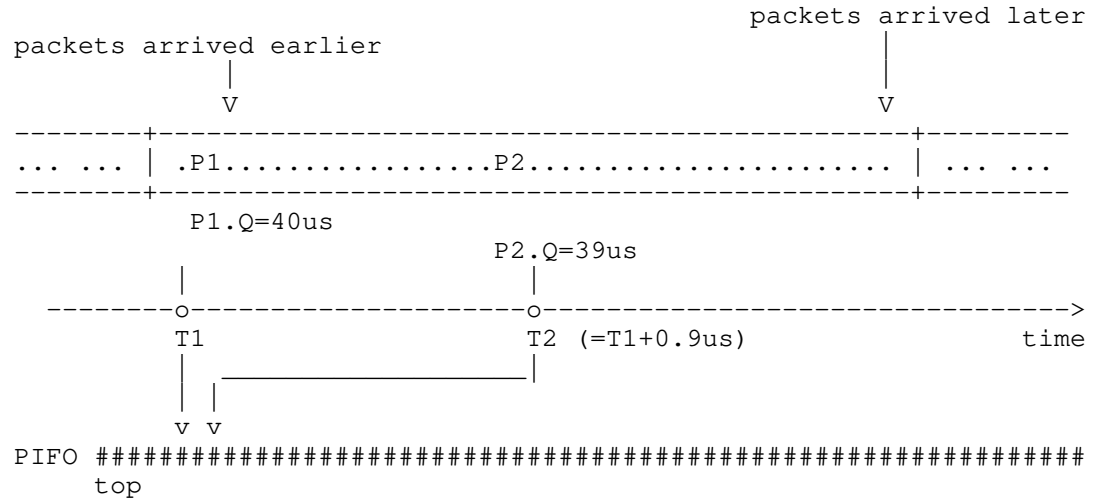

Figure 5: Disorder Illustration of PIFO

DetNet architecture [RFC8655] provides Packet Ordering Function (POF), that can be used to solve the above disorder problem caused by the latency compensation.

Alternatively, Section 8 provides E|D decoupling method to firstly absorb latency deviation E by the pre-scheduler which may maintain FIFO queue per incoming port plus delay level. In this case, packets from the same flow will only determine the damping delay, but not the position, in the FIFO based on latency deviation E, to avoid disorder. Latency deviation E no longer works in the post-scheduler.

Note that in practical situations, two back-to-back packets of the same flow are generally evenly distributed within the burst interval by regulation, which means that the distance between these two packets is generally much greater than the calculation accuracy mentioned above, meaning that the disordered phenomenon will not really occur. For example, the regulated result meets a Length Rate Quotient (LRQ) constraint, and the time interval between two consecutive packets of size l_i and l_j should be at least l_i/r, where r is the flow rate (i.e., the reserved bandwidth of the flow). This can be done by LRQ based regulation, or enhanced leaky bucket based regulation, depending on implementation.

7.4. Option-4: Latency Compensation plus RPQ

As shown in Figure 6, a received packet is inserted to the appropriate RPQ queue with specific CT to meet CT <= Q < CT+CTI when the packet arrived at the scheduler, where Q = D + E - F. Depending on the situation of the accumulated burst arrived at the input port, different packets may have different input latency deviation E. Latency compensation will convert the input ineligible arrivals pattern (if possible) into an eligible arrivals pattern.

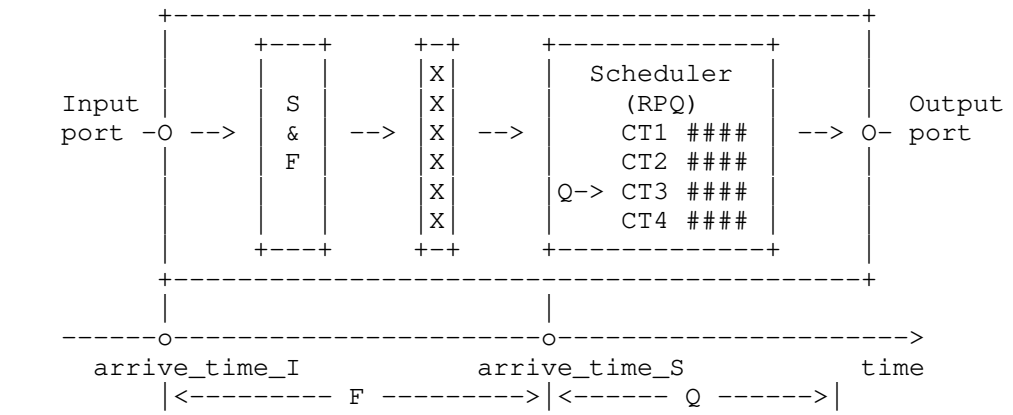

Figure 6: Latency Compensation plus RPQ

The planned residence time (D) and latency deviation (E) should be carried in the packet.

The enqueue and dequeue operations are the same as Section 7.2.

In this option the dequeue rule of on-time mode can guarantee jitter with the help of factor E to absorb jitter per hop. See Section 8 for more information.

Figure 7 depicts an example of packets inserted to the RPQ queues.

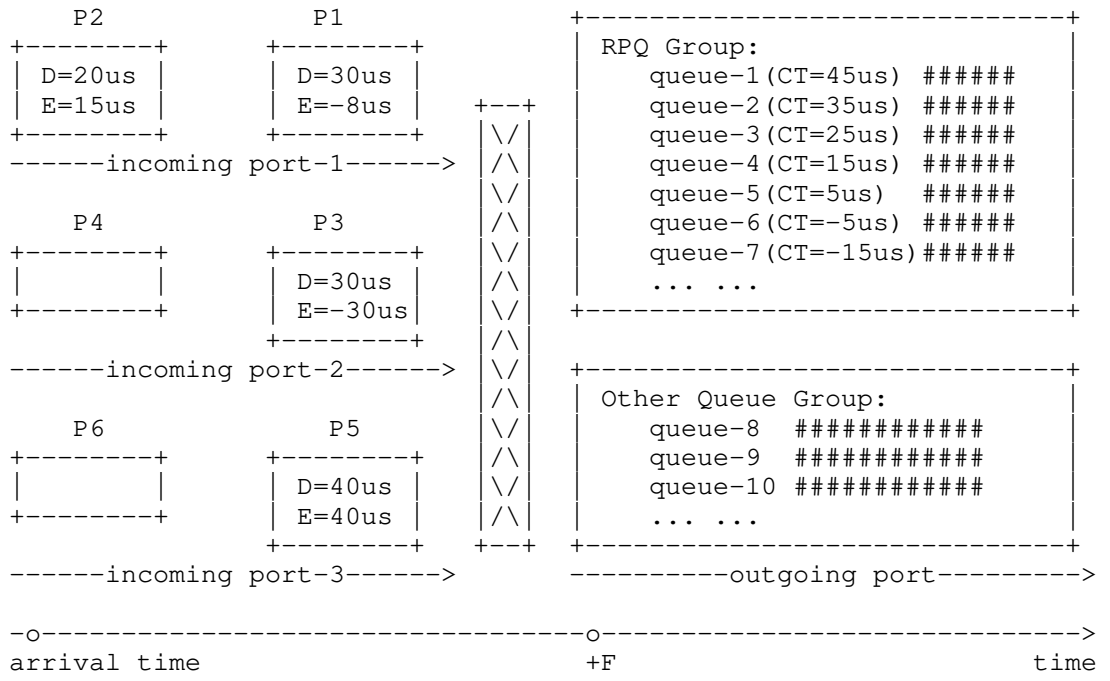

Figure 7: Time Sensitive Packets Inserted to RPQ

As shown in Figure 7, the node successively receives six packets from three incoming ports, among which packet 1, 2, 3 and 5 have corresponding deadline information, while packet 4 and 6 are best-effort packets. These packets need to be forwarded to the same outgoing port. It is assumed that they arrive at the line card where the outgoing port is located at almost the same time after the forwarding delay (F = 5us). At this time, the queue status of the outgoing port is shown in the figure. Then:

* The allowable queueing delay (Q) of packet 1 is 30 - 8 - 5 = 17us, and it will be put into queue-4 (its CT is 15us), meeting the condition that Q is in the range [15, 25).

* The allowable queueing delay (Q) of packet 2 is 20 + 15 - 5 = 30us, and it will be put into queue-3 (its CT is 25us), meeting the condition that Q is in the range [25, 35).

* The allowable queueing delay (Q) of packet 3 is 30 - 30 - 5 = -5us, and it will be put into queue-6 (its CT is -5us), meeting the condition that Q is in the range [-5, 5).

* The allowable queueing delay (Q) of packet 5 in the node is 40 + 40 - 5 = 75us, and the queue it is placed on is not shown in the figure (such as a hierarchical queue).

* Packets 4 and 6 will be put into the non-deadline queue in the traditional way.

According to Section 4.3, An eligible packet (i.e., E = 0) from a specific delay level, even at the end of the inserted queue, can ensure that it does not exceed its deadline, which is the key role of the CTI factor in the condition equation. Now, assuming that a packet is penalized to a lower priority queue based on its positive E, this penalty will not result in more than expected delay, apart from potential delay E.

For example, when a packet is inserted queue based on

$$CT_x <= Q < CT_x + CTI$$

even if it is at the end of the queue, according to D = Q - E, i.e., after time E (the penalty time), we have

$$CT_x - E <= Q - E < CT_x - E + CTI$$

That is

$$CT_y <= D < CT_y + CTI$$

So, in essence, it is still equivalent to an eligible packet entering the corresponding queue based on its delay level, and apply the schedulability condition.

7.4.1. Packet Disorder Considerations

Suppose that two packets, P1, P2, are generated instantaneously from a specific flow at the source, and the two packets have the same planned residence time. P1 may face less interference delay than P2 in their journey. When they arrive at an intermediate node in turn, P2 will have less allowable queueing delay (Q) than P1 to try to stay close to P1 again. It should be noted that to compare who is earlier is based on queue's CT and packet's Q, according to the above queueing rule (CT <= Q < CT+CTI), and the CT of the queue is not changed in real-time, but gradually with the decreasing step RTI. It is possible to get an unexpected comparison result.

As shown in Figure 8, P1 and P2 are two packets belonging to the same flow. The arrival time when they are received on the scheduler is shown in the figure. Suppose that CTI is 10us, the decreasing step

RTI is 1us, and the transmission time of each packet is 0.01us. Also suppose that the Q values of two adjacent packets P1 and P2 are 40us and 39us respectively, and they are both received in the window from T0 to T0+1us. P1 will enter queue-B with CT range [40, 50), while P2 will enter queue-A with CT range [30, 40) just before the rotation event occurred. This means that P2 will be scheduled before P1, resulting in disorder.

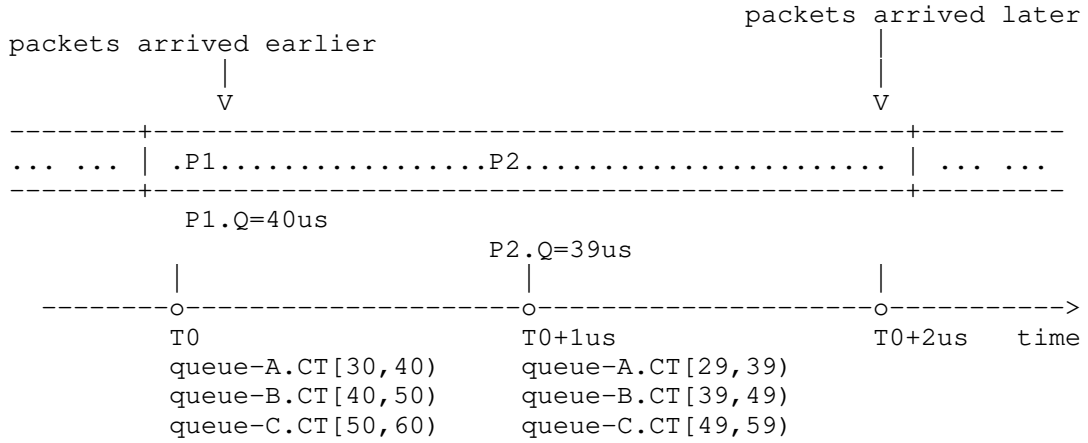

Figure 8: Disorder Illustration of RPQ

DetNet architecture [RFC8655] provides Packet Ordering Function (POF), that can be used to solve the above disorder problem caused by the latency compensation.

Alternatively, Section 8 provides E|D decoupling method to firstly absorb latency deviation E by the pre-scheduler which may maintain FIFO queue per incoming port plus delay level. In this case, packets from the same flow will only determine the damping delay, but not the position, in the FIFO based on latency deviation E, to avoid disorder. Latency deviation E no longer works in the post-scheduler.

Note that in practical situations, two back-to-back packets of the same flow are generally evenly distributed within the burst interval by policing, which means that the distance between these two packets is generally much greater than the calculation accuracy mentioned above, meaning that the disordered phenomenon will not really occur. For example, the regulated result meets a Length Rate Quotient (LRQ) constraint.

8. Jitter Performance by On-time Scheduling

The enqueue and dequeue rule of on-time mode described in Section 7.3 and Section 7.4 will absorb latency deviation E on each hop, and achieve a low jitter. The ultimate E2E jitter depends on the delay experienced on the last node of the flow, which may be from 0 to the delay bound, i.e., the corresponding delay level d_i.

Depending on different methods of absorbing E, there are slight differences in scheduling behavior.

* E+D integration: E will be added to D to get the adjusted D'; The packet is scheduled by the EDF scheduler configured with on-time mode based on D'.

* E|D decoupling: There are 2-tier schedulers; The packet is scheduled by pre-scheduler configured with on-time mode based on E, then scheduled by post-scheduler configured with in-time mode based on D.

In the case of E+D integration, it may explicitly introduce the mandatory hold time, and cause that the actual departure time of the packet may be after its deadline. Assuming that the eligible arrivals pattern of all delay levels causes the scheduler to work at full speed (i.e., service rate C), then for in-time mode, the worst case is that there may be a packet of a specific delay level to be sent just before its deadline during the busy period; While for E+D integration case, the busy period may just start at its deadline and cause the sending time of the packet to exceed its deadline. However, as mentioned above, the worst case of this exceeding value will not exceed the delay level value, which is intuitive because it is equivalent to the situation where the observed packet arrives asynchronously after the delay level value. Note that this exceeding deadline does not accumulate with the number of hops. The E2E latency is in the range [D*hops, D*hops+d_i].

In the case of E|D decoupling, the explicitly mandatory hold time is only contributed by E ensured by the pre-scheduler (configured with on-time mode), and the actual departure time (from the post-scheduler) of the packet will always be before its deadline. Assuming that the eligible arrivals pattern of all delay levels cause the post-scheduler (configured with in-time mode) to work at full speed, for E|D decoupling case, the worst case is that there may be a packet of a specific delay level to be sent just before its deadline during the busy period. The E2E latency is in the range [D*(hops-1), D*(hops-1)+d_i]. Note that the pre-scheduler may maintain a PIFO, an RPQ, or several FIFO queues each for particular "incoming port + delay level". Figure 9 shows the functional entities inside the node.

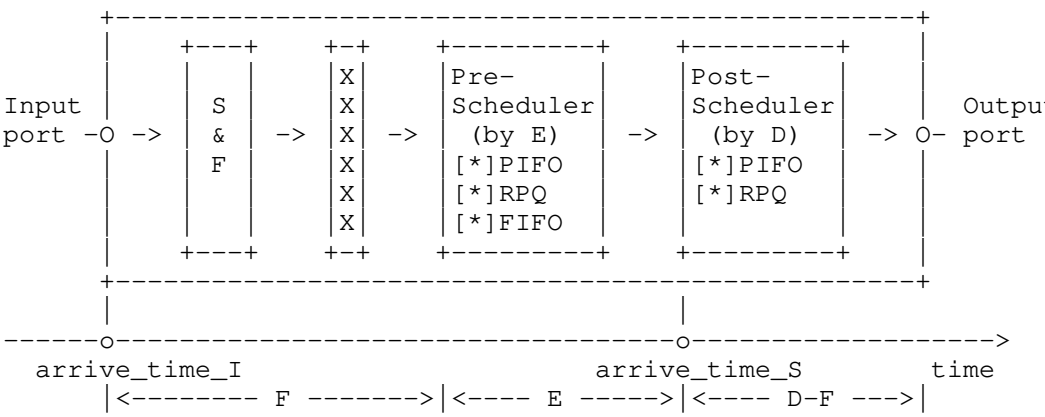

Figure 9: E|D decoupling with 2-tier Schedulers

The following Figure 10 shows the difference between on-time scheduling and in-time scheduling.

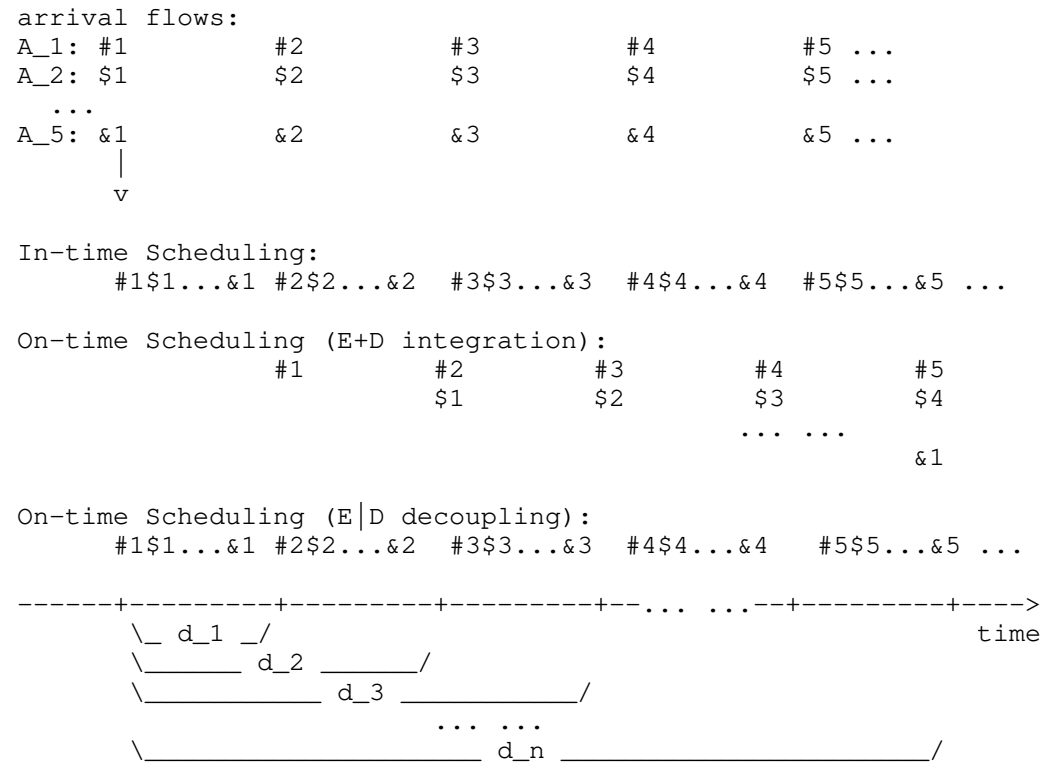

Figure 10: Difference between In-time and On-time Scheduling

As shown in the figure, each burst of A_1 (corresponding to delay level d_1) is termed as #num, each burst of A_2 (corresponding to delay level d_2) as $num, and each burst of A_5 (corresponding to delay level d_5) as &num. A single burst may contain multiple packets. For example, burst #1 may contain several packets, and the actual time interval between #1 and #2 may be small. Although the figure shows the example that the burst interval of multiple flows is the same and the phase is aligned, the actual situation is far from that. However, this example depicts the typical scheduling behavior.

In the in-time scheduling, all concurrent traffic of multiple levels will be scheduled as soon as possible according to priority, to construct a busy period. For example, in the duration d_1, in addition to the burst #1 that must be sent, the burst $1~&1 may also be sent, but the latter is not necessarily scheduled to be sent before the burst #2 as shown in the figure. Here we clearly see that in-time scheduling cannot guarantee jitter.

While in the case of on-time scheduling with E+D integration option, each burst is scheduled at its deadline, which may just be the begin of the busy period. Because of the scheduling delay, the transmission of the burst will exceed its deadline. The last packet of the burst will face more delay than the first packet. For example, when burst #5 enters the PIFO, it may have the same deadline with bursts from $4 of A_2 to &1 of A_5. When the deadlines of multiple packets are the same, use planned residence time (D) as tiebreaker, i.e., the smaller the D, the smaller the rank. So, #5 send first and may exceed the deadline by one d_1; Then send $4 and may exceed the deadline by one d_2; ...; Finally, send &1 and may exceed the deadline by one d_5.

In the case of on-time scheduling with E|D decoupling, assuming that the latency deviation E for each burst is 0 in the above figure, all concurrent traffic of multiple levels, similar to in-time, will also be scheduled as soon as possible according to priority, to construct a busy period. For example, in the duration d_1, in addition to the burst #1 that must be sent, the burst $1~&1 may also be sent. However, $1~&1 will obtain punishment based on their E on the next node (not shown in the figure).

9. Resource Reservation

Generally, a path may carry multiple DetNet flows with different delay levels. For a certain delay level d_i, the path will reserve some resources from the delay resource pool of the link. The delay resource pool here, as leaky bucket constraint function shown in Section 3.2.1 or Section 4.3.2, is a set of preset parameters that meet the schedulability conditions. For example, the level d_1 has a burst upper limit of b_1 and a bandwidth upper limit of r_1. A path j may allocate partial resources (b_i_j, r_i_j) from the resource pool (b_i, r_i) of the link's delay level d_i. A DetNet flow k that carried in path j, may use resources (b_i_j_k, r_i_j_k) according to its T_SPEC. It can be seen that the values of b_i_j and r_i_j determine the scale of the number of paths that can be supported, while the values of b_i_j_k and r_i_j_k determine the scale of the number of flows that can be supported. The following expression exists.

* sum(b_i_j_k) <= b_i_j, for all flow k over the path j.

* sum(r_i_j_k) <= r_i_j, for all flow k over the path j.

* sum(b_i_j) <= b_i, for all path j through the specific link.

* sum(r_i_j) <= r_i, for all path j through the specific link.

9.1. Delay Resource Definition

The delay resources of a link can be represented as the corresponding burst and bandwidth resources for each delay level. Basically, what delay levels (e.g., 10us, 20us, 30us, etc) are supported by a link should be included in the link capability.

Figure 11 shows the delay resource model of the link. The resource information of each delay level includes the following attributes:

* Delay Bound: Refers to the delay bound intra node corresponding to this delay level. It is a pre-configuration value.

* Maximum Reservable Bursts: Refers to the maximum amount of bit quota corresponding to this delay level. It is a pre-allocated value or resource limit set based on the schedulability condition.

* Utilized Bursts: Refers to the burst utilization of this delay level.

* Maximum Reservable Bandwidth: Refers to the maximum amount bandwidth corresponding to this delay level. It is a pre-allocated value or resource limit set based on the schedulability condition.

* Utilized Bandwidth: Refers to the bandwidth utilization of this delay level.

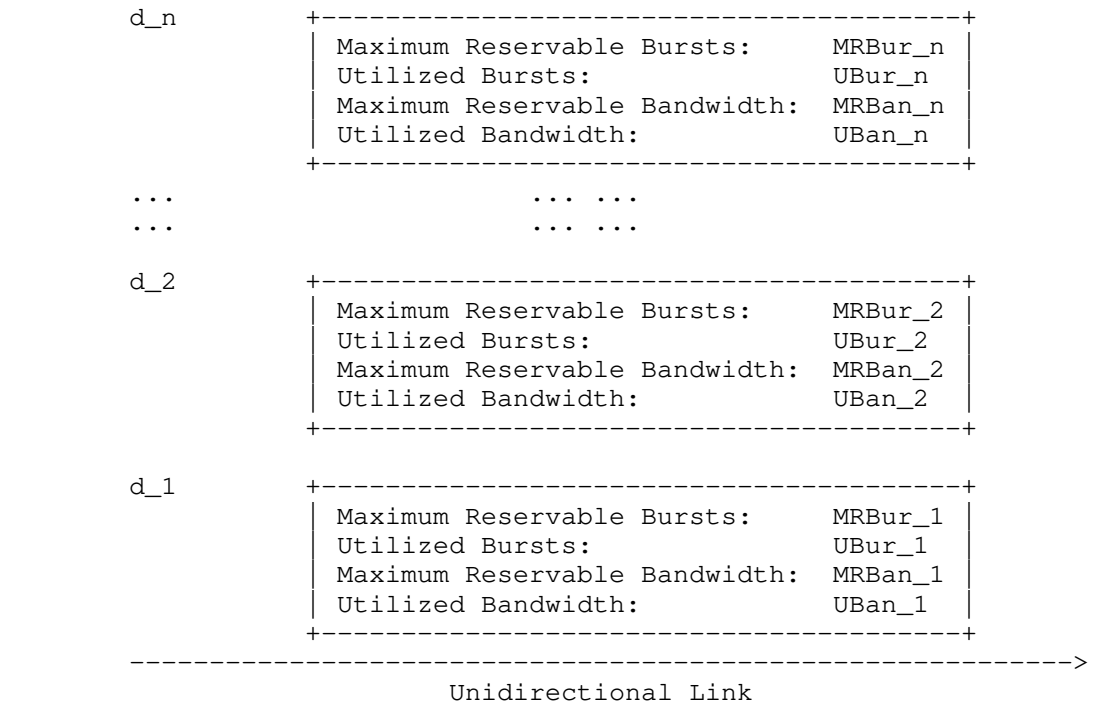

Figure 11: Delay Resource of the Link

For a specific link:

* If Maximum Reservable Bursts and Maximum Reservable Bandwidth are used for schedulability condition checking, they need to set reasonable values at the beginning to meet the schedulability condition, and in the future, there is no need to execute schedulability condition checking during the setup procedure of any flow passing through this link, but only need to check that the aggregated burst and bandwidth of all flows belonging to the same delay level do not exceed Maximum Reservable Bursts and Maximum Reservable Bandwidth, respectively.

* If Utilized Bursts and Utilized Bandwidth are used for schedulability condition checking, there is necessary to execute schedulability condition checking during the setup procedure of any new flows passing through this link.

The IGP/BGP extensions to advertise the link's capability and delay resource is defined in [I-D.peng-lsr-deterministic-traffic-engineering].

9.2. Traffic Engineering Path Calculation

A candidate path may be selected according to the end-to-end delay requirement of the flow. Subtract the accumulated link propagation delay from the end-to-end delay requirement, and then divide it by the number of hops to obtain the average planned residence time (D) for each node. Or, different nodes may have different planned residence time (D). By default, select the appropriate delay level d_i (d_i <= D-F) closest to the planned residence time (D), and then reserve resources from delay level d_i on each hop. A local policy may allow more larger D to consume resources with smaller delay levels.

Note that it is planned residence time (D), not delay level (d_i), carried in the forwarding packets.

9.3. Overprovision Analysis

For each delay level d_i, the delay resource of the specific link is (b_i, r_i). A path j may allocate partial resources (b_i_j, r_i_j) from the resource pool (b_i, r_i). In order to support more d_i flows in the network, it is necessary to set larger b_i and r_i. However, as mentioned earlier, the values of b_i and r_i are set according to schedulability conditions and cannot be set at will.

For bandwidth resource reservation case, the upper limit of the total bandwidth that can be reserved for all aggregated flows of delay level d_i is r_i, which is the same as the behavior of traditional bandwidth resource reservation. There is no special requirement for the measurement interval of calculating bandwidth value.

For the burst resource reservation case, the upper limit of the total burst that can be reserved for all aggregated flows of delay level d_i is b_i. If the burst of each flow of level d_i is b_k, then the number of flows can be supported is b_i/b_k, which is the worst case considering the concurrent arrival of these flows. However, the burst resource reservation is independent of bandwidth resource, i.e., it does not take the calculation result of b_k/d_i to get an overprovision bandwidth and then to affect the reservable bandwidth resources.

By providing multiple delay levels, we can allocate 100% of the link bandwidth to DetNet flows, as can be seen from the schedulability condition equation.

10. Policing on the Ingress

On the ingress PE node, policing must be performed on the incoming port, so that DetNet flow does not violate its T-SPEC. This kind of traffic regulation is usually the shaping using leaky bucket. After policing, the shaped pattern of the DetNet flow may contain discrete multiple bursts evenly distributed within its periodic service burst interval (SBI). For example, An arriving elephant flow will be diluted and released to the EDF scheduler.

According to [RFC9016], the values of Burst Interval, MaxPacketsPerInterval, MaxPayloadSize of the DetNet flow will be written in the SLA between the customer and the network provider, and the network entry node will set the corresponding bucket depth according to MaxPayloadSize to forcibly delay the excess bursts. The entry node also sets the corresponding bucket rate according to the promised arrival rate.

The shaped pattern is generally inconsistent with the original arrival pattern of the DetNet flow, and some bursts of the original arrival pattern may experience more shaping delay than others. The shaped pattern and the original arrival pattern can be as consistent as possible by increasing the bucket depth, but this means that the flow will occupy more burst resources, and reduce the service scale that the network can support according to the schedulability conditions.

On the network entrance node, for the burst with applied shaping delay, shaping delay cannot be included in the latency compensation equation, otherwise, it will make that burst catch up with the previous burst, resulting in damage to the policing result and violation of the arrival constraint function. Please refer to [I-D.peng-detnet-policing-jitter-control] for the elimination of jitter caused by shaping delay on the network entrance node.

Then, the regulated traffic arrives at the EDF scheduler on the outgoing port. Since the traffic of each delay level meets the leaky bucket arrival constraint function and the parameters of the shaping curve do not exceed the limits of the parameters provided by the schedulability conditions, the traffic can be successfully scheduled based on deadline.

Note that the flow arrived at the next hop, after reshaping or latency compensation, will still follow the arrival constraint function of that flow. When this flow is aggregated with other flows and sent to the same outgoing port, within any duration d_i, the aggregated d_i traffic will not exceed the burst and bandwidth resources of delay level d_i reserved by these flows on the outgoing port.

Figure 12 depicts an example of policing and deadline based scheduling on the ingress PE node in the case of option-4 with on-time mode. In the figure, the shaping delay of each burst is termed as S#.

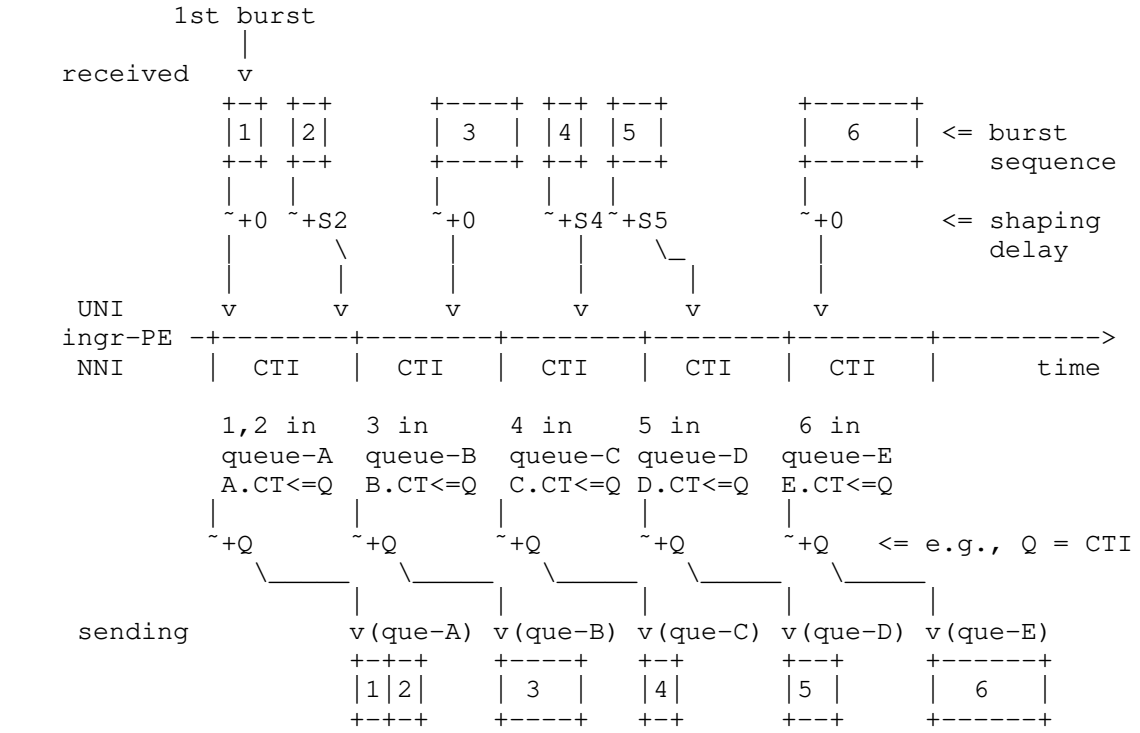

Figure 12: Deadline Based Packets Orchestrating

There are 6 bursts received from the client. The burst-2, 4, 5 has policing delay S2, S4, S5 respectively, due to the consumption of tokens by previous burst. While burst-1, 3, 6 has zero policing delay because the number of tokens is sufficient. The policing makes 6 bursts roughly distributed within the service burst interval.

Assuming that the forwarding delay F experienced by all bursts is 0. In the case of latency compensation plus RPQ, they will have the same allowable queueing delay (Q), regardless of whether they have experienced policing delay before. When the packets of burst-1, 2 arrive at the scheduler, according to CT <= Q < CT+CTI, they will be placed in Queue-A with matched CT and waiting to be sent. Similarly, when the packets of burst-3/4/5/6 arrive at the scheduler, they will be placed in Queue-B/C/D/E respectively and waiting to be sent according to the de-queue rules of on-time mode. Note that each sending burst may get a latency deviation E, especially for burst-2, which is sent closely adjacent to burst-1 in the sending pattern.

11. Compatibility with Legacy Device

Deadline is suitable for end-to-end and interconnection between different networks. A large-scale network may span multiple networks, and one of the goals of DetNet is to connect each network domain to provide end-to-end deterministic delay service. The adoption techniques and capabilities of each network are different, and the corresponding topology models are either piecewise or nested.

For a particular path, if only some nodes in the path upgrade support the deadline based mechanism defined in this document, the end-to-end deterministic delay/jitter target will only be partially achieved. Those legacy devices may adopt the existing SP or WFQ mechanisms, and ignore the possible deadline information carried in the packet, thus the residence delay produced by them cannot be perceived by the adjacent upgraded node. The more upgraded nodes included in the path, the closer to the delay/jitter target. Although, the legacy devices may not support the data plane mechanism described in this document, but they can be freely programmed (such as P4 language) to measure and insert the deadline information into packets, in this case the delay/jitter target may be achieved.

Only a few key nodes are upgraded to support deadline mechanism, which is low-cost, but can meet a flow with relatively loose time sensitive. Figure 13 shows an example of upgrading only several network border nodes. In the figure, only R1, R2, R3 and R4 are upgraded to support deadline based mechanism. A deterministic path across domain 1, 2, and 3 is established, which contains nodes R1, R2, R3, and R4, as well as explicit nodes in each domain. Domain 1, 2 and 3 use the traditional SP mechanism. The encoding of the packet sent by R1 includes the planned residence time and the latency deviation E. Especially, DS filed in IP header ([RFC2474]) are also set to appropriate values. The basic principle of setting is that the less the planned residence time, the higher the priority, to avoid the interference by non DetNet flows.

The delay analysis based on strict priority without re-shaping in each domain can be found in [SP-LATENCY], which gives the equation to evaluate the worst-case delay of each hop. The worst-case delay per hop depends on the number of hops and the burst size of interference flows that may be faced on each hop. [EF-FIFO] also shows that, for FIFO packet scheduling be used to support the EF (expedited forwarding) per-hop behavior (PHB), if the network utilization level alpha < l/(H-l), the worst-case delay bound is inversely proportional to 1-alpha*(H-1), where H is the number of hops in the longest path of the network. Having fewer hops in the SP domain is better.

Although the EDF scheduling with in-time mode, the SP scheduling and EF FIFO scheduling are all work-conserving, the EDF scheduling can further distinguish between urgent and non urgent packets according to deadline information other than traffic class. The operation of dynamically modifying the key fields, i.e., the latency deviation (E), of the packet can avoid always overestimating worst-case latency on all hops just like SP.

For a specific DetNet flow, if it experiences too much latency in the SP domain (due to unreasonable setting of DS field and the inability to distinguish between DetNet and non DetNet flows), even if the border node accelerates the transmission, it may not be able to achieve the target of low E2E latency. If the traffic experiences less latency within the SP domain, the on-time scheduling mode applied on the border node can help achieve the end-to-end jitter target.

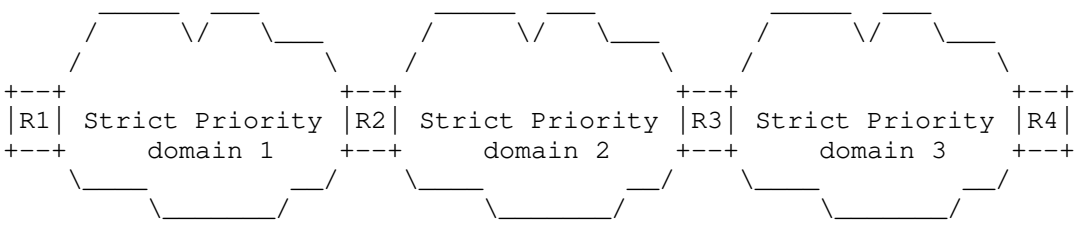

Figure 13: Example of partial upgrade

12. Deployment Considerations

According to the above schedulability conditions, each delay level d_i has dedicated delay resources, and the smaller d_i, the more valuable it is. The operator needs to match the corresponding d_i for each flow. It should be noted that the per-hop latency provided by EDF for the flow is based on flow's RSpec, not TSpec.

In the case of option-3 and 4 with in-time scheduling behavior, more buffer is required to absorb burst accumulation.

For a specific flow, the accumulated bursts on a intermediate node consists of multiple rounds of burst interval. For example, the packets generated by the source within the first round of burst interval (always experiencing the worst case delay along the path) is caught up by the packets generated within the second round of burst interval (always experiencing the best case delay along the path). For delay level d_i, the worst case delay is d_i, the best case delay is l/R, where l is the smallest packet size of the flow, R is the port rate. For simplicity to get the estimate size of accumulated bursts, here we just take the best case delay as 0. Drawing on the method provided in [SP-LATENCY], the accumulated bursts of d_i is:

* ACC_BUR_i = ((d_i * h) / burst_interval) * b_i

For example, d_i is 10 us, burst_interval is 250 us, this means that within the 25th hop, there will only be one b_10 burst in the queue. If it exceeds 25 hops and is within 50 hops, there may be two b_10 burst simutaneously in the queue.

The accumulated bursts of other delay levels can be similarly estimated. Operators need to evaluate the required buffer size based on network hops and the supported delay levels. The benefit of in-time scheduling is to obtain an E2E latency of no more than D*hops as small as possible.

Operators may also apply on-time scheduling per hop to simplify the design of buffers. On-time scheduling absorbed latency deviation E on each hop and can get a jitter for each delay level to the value of delay level in theory (i.e., the worst case is that on the last node there are full flows contributed by all delay levels that are discharging floodwater at the same time, however, in reality, the DetNet flow of the output port facing the destination customer side may only involve one delay level, then the jitter may be only one CTI (e.g., 10us)).

In summary, the in-time scheduling with latency compensation, can suffer from the uncertainty caused by burst accumulation, and it is recommended only deployed in small networks, i.e., a limited domain with a small number of hops, where the burst accumulation issue is not serious; The on-time scheduling per hop is recommended to be used in large networks.

On-time scheduling has an additional cost of pre-scheduler component compared to in-time scheduling. Operators may enable in-time scheduling on intermediate devices and enable on-time scheduling on network exit devices to achieve the goal of low jitter of EDF path. In this case, the local policy of the intermediate device should allow the use of in-time scheduling for the packets that actually require on-time service.

13. Evaluations

Now we summarize how the deadline based mechanism ensures bounded latency and jitter as below:

1) Partition delay resource for each delay level on the outgoing port according to the schedulability condition, i.e., preset parameters of the arrival constraint function.

2) Reserve delay resource on each link of the calculated path. This step is also known as admission condition check.

3) Execute policing on the network entry, to let the admitted traffic obey its constraint function (i.e., TSpec).

4) Execute reshaping or latency compensation (recommended) in the network core for each flow, to convert the ineligible arrivals to eligible arrivals that still obey the constraint function of each flow.

5) Guarantee bounded delay by in-time scheduling mode; Guarantee bounded delay and jitter by on-time scheduling mode.

13.1. Large Scaling Requirements Matching Degree

The following table is the evaluation results based on the requirements that is defined in [I-D.ietf-detnet-scaling-requirements]. Note that all asynchronous mechanisms (such as EDF, ATS) do not require complete synchronization of crystal oscillator frequencies between devices. The latency error caused by the deviations of clocks from their nominal rates, e.g., +100ppm, is generally in the nanosecond range and can be ignored.

requirements	Evaluation	Notes
3.1 Tolerate Time Asynchrony	Yes	No time sync needed; No frequency sync needed.
3.2 Support Large Single-hop Propagation Latency	Yes	The eligible arrival of flows is independent with the link propagation delay.
3.3 Accommodate the Higher Link Speed	Partial	The higher service rate, the more buffer needed for each delay level. And, extra instructions to calculate E.
3.4 Be Scalable to the Large Number of Flows and Tolerate High Utilization	Yes	Limited number of delay levels are mapped by lots of flows. No overprovision in the resource reservation. Utilization may reach 100% link bandwidth. The unused bandwidth of the high delay level can be used by the low levels or BE flows.
3.5 Tolerate Failures of Links or Nodes and Topology Changes	N/A	Independent of queueing mechanism.
3.6 Prevent Flow Fluctuation	Yes	Flows are permitted based on the resources reservation of delay levels, and isolated from each other.
3.7 Be scalable to a Large Number of Hops with Complex Topology	Yes	E2E latency is liner with hops, from ultra-low to low latency by multiple delay levels. E2E jitter is low by on-time scheduling.
3.8 Support Multi-Mechanisms in Single Domain and Multi-Domains	N/A	Independent of queueing mechanism.

Figure 14: Evaluation for Large Scaling Requirements

13.2. Taxonomy Considerations

[I-D.ietf-detnet-dataplane-taxonomy] provides criteria for classifying data plane solutions.

For performance, the per hop latency dominant factor of EDF is the delay levels that is defined according to schedulability condition.

For functional characteristics, EDF is non-periodic, class level for traffic granularity, and right-bounded or bounded for time Bounds.

* Non-periodic: The scheduling power of an EDF is measured over an arbitrarily long non repetitive time range, scheduling in an orderly manner according to the urgency of the packets, and there is no defined periodic quantification unit of scheduling power.

* Class level: DetNet Flows can be grouped by similar service requirements, i.e., delay levels provided in the network. Packets will be provided EDF service based on delay level, without checking flow identification.

* Right-bounded/bounded: A packet's deadline is defined as its maximum time bound. So EDF with in-time mode is right-bounded. While EDF with on-time mode, due to further limiting the minimum time bound, is bounded.

[I-D.ietf-detnet-dataplane-taxonomy] also specifies the suitable categories of solutions for DetNet. According to the above functional characteristics, EDF with in-time mode will map to right-bounded category, and EDF with on-time mode will map to class level non-periodic bounded category.

13.3. Examples

This section describes the example of how the deadline mechanism supports DetNet flows with different latency requirements.

13.3.1. Heavyweight Loading Example

This example observes the service scale and different latency bound supported by the deadline mechanism in the heavyweight loading case.

Figure 15 provides a typical reference topology that serves to represent or measure the multihop jitter and latency experience of a single "flow i" across N hops (in the figure, N=10). On each of the N outgoing interfaces (represented by circles in the figure), "flow

i" has to compete with different flows (represented by different symbols on each hop). Especially, the competed flows arrive simultaneously at multiple incoming ports, with the same starting time when measuring their respective residence time. The characteristic of this reference topology is that every link that "flow i" passes through may be a bottleneck link with 100% network utilization, causing "flow i" to achieve the worst-case latency on each hop.

As shown in Figure 15:

* Network transmission capacity: each link has rate 10 Gbps. Assuming the service rate of EDF scheduler allocate the total port bandwidth.

* TSpec of each flow, maybe:

  - burst size 1000 bits, and average arrival rate 1 Mbps.

  - or, burst size 1000 bits, and average arrival rate 10 Mbps.

  - or, burst size 1000 bits, and average arrival rate 100 Mbps.

* RSpec of each flow, maybe:

  - E2E latency 100us, and E2E jitter less than 10us or 100us.

  - or, E2E latency 200us, and E2E jitter less than 20us or 200us.

  - or, E2E latency 300us, and E2E jitter less than 30us or 300us.

  - or, E2E latency 400us, and E2E jitter less than 40us or 400us.

  - or, E2E latency 500us, and E2E jitter less than 50us or 500us.

  - or, E2E latency 600us, and E2E jitter less than 60us or 600us.

  - or, E2E latency 700us, and E2E jitter less than 70us or 700us.

  - or, E2E latency 800us, and E2E jitter less than 80us or 800us.

  - or, E2E latency 900us, and E2E jitter less than 90us or 900us.

  - or, E2E latency 1ms, and E2E jitter less than 100us or 1ms.

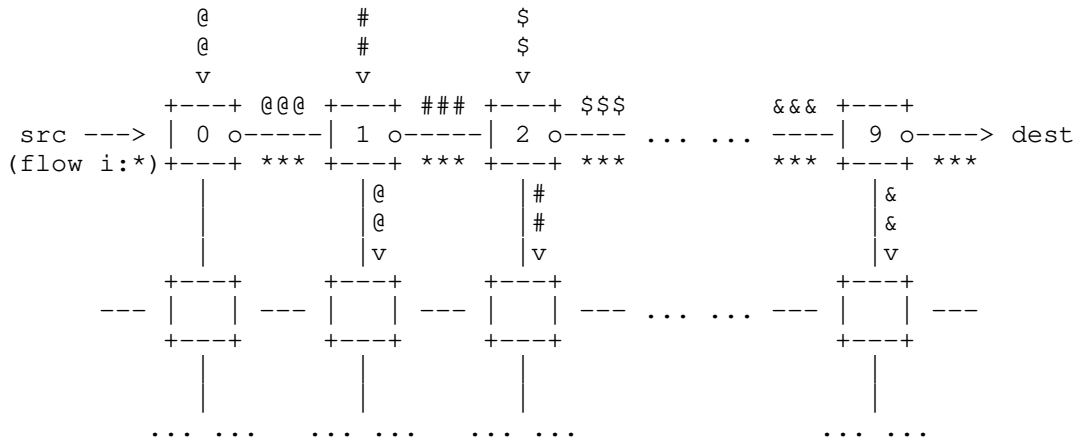

Figure 15: Heavyweight Loading Topology Example

For the observed flow i (marked with *), its TSpec and RSpec may be any of the above. Assuming that the path calculated by the controller for the flow i passes through 10 nodes (i.e., node 0~9). Especially, at each hop, flow i may conflict with other deterministic flows, also with similar TSpec and RSpec as above, originated from other sources, e.g., conflicts with flow-set "@" at node 0, conflicts with flow-set "#" at node 1, and so on.

For each link along the path, it may provide multiple delay levels, e.g., d1 (10us), d2 (20us), ..., d10 (100us). Assuming no link propagation delay and intra node forwarding delay. If flow i uses d1 resources, it can ensure an E2E latency of 100us (i.e., d1 * 10 hops), and jitter of 10us(on-time mode) or 100us (in-time mode). The results of using resources of other delay levels are similar.

The table below shows the possible tight allocation of delay resources on each link based on Equation-1, as well as the corresponding service scale supported, where, b = utilized burst resource (K bits), r = utilized bandwidth resource (Mbps), s = service scale (number), assuming that the resource limit of each delay level is b_limit = 100000 bits, r_limit = 1 Gbps.

Note that in the table each row only shows the data where all flows served by all delay levels have the same TSpec (e.g., in row-1, TSpec per flow is burst size 1000 bits and arrival rate 1 Mbps), while in reality, flows served by different delay levels generally have different TSpec. It is easy to add rows to describe various combinations.

		d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
row-1	b	100	99	98	97	96	95	94	93	92	91
TSpec: 1000 bits 1 Mbps	r	100	99	98	97	96	95	94	93	92	91
	s	100	99	98	97	96	95	94	93	92	91
row-2	b	100	90	81	73	66	60	53	48	43	39
TSpec: 1000 bits 10 Mbps	r	1000	900	810	729	656	590	531	478	430	387
	s	100	90	81	72	65	59	53	47	43	38
row-3	b	100	90	80	70	60	50	40	30	20	10
TSpec: 1000 bits 100 Mbps	r	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
	s	10	10	10	10	10	10	10	10	10	10

Figure 16: Delay Resource Pool and Service Scale Example

13.3.2. Lightweight Loading Examples

The following examples observe how the preset service scale is supported and mapped to different delay levels by the deadline mechanism in the lightweight loading case.

In these examples, the network only contains a small number of bottleneck links with low network utilization, and it can be considered as the lightweight loading case of Figure 15. Lightweight loading usually means having a smaller calculated worst-case latency per hop, or the actual latency experienced doesn't reach the worst-case latency.

13.3.2.1. Grid Reference Topology

[I-D.ietf-detnet-dataplane-taxonomy] describes a Grid topology (Figure 17) with partial mesh. Three flow types, i.e., audio, video, and CC (Command and Control) are considered to require deterministic networking services. Among them, audio and CC flows consume less bandwidth (1.6 Mbps per flow and 0.48 Mbps per flow respectively) but both require lower E2E latency (5ms), while video flows consume more bandwidth (11 Mbps per flow) but can tolerate larger E2E latency (10ms).

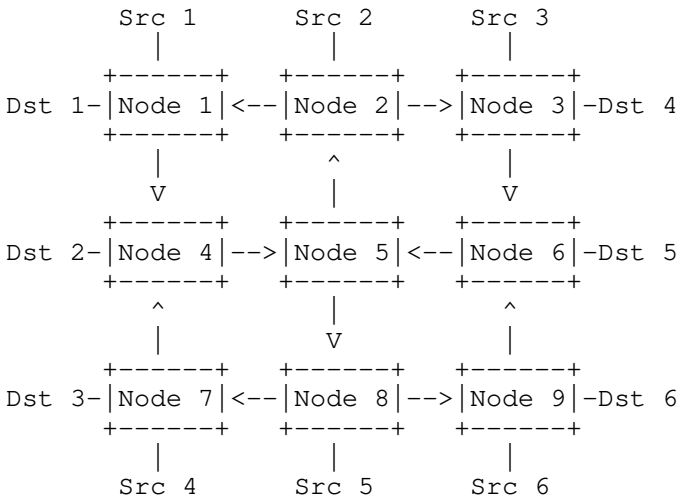

Figure 17: Grid Reference Topology

According to the preset rules that generate a unique route for every source and destination pair, the details of all paths are as follows:

Src1-1-Dst1
Src1-1-4-Dst2
Src1-1-4-5-8-7-Dst3
Src1-1-4-5-2-3-Dst4
Src1-1-4-5-8-9-6-Dst5
Src1-1-4-5-8-9-Dst6

Src2-2-1-Dst1
Src2-2-1-4-Dst2
Src2-2-3-6-5-8-7-Dst3
Src2-2-3-Dst4
Src2-2-3-6-Dst5
Src2-2-3-6-5-8-9-Dst6

Src3-3-6-5-2-1-Dst1
Src3-3-6-5-2-1-4-Dst2
Src3-3-6-5-8-7-Dst3
Src3-3-Dst4
Src3-3-6-Dst5
Src3-3-6-5-8-9-Dst6

```
Src4-7-4-5-2-1-Dst1
Src4-7-4-Dst2
Src4-7-Dst3
Src4-7-4-5-2-3-Dst4
Src4-7-4-5-8-9-6-Dst5
Src4-7-4-5-8-9-Dst6

Src5-8-7-4-5-2-1-Dst1
Src5-8-7-4-Dst2
Src5-8-7-Dst3
Src5-8-7-4-5-2-3-Dst4
Src5-8-9-6-Dst5
Src5-8-9-Dst6

Src6-9-6-5-2-1-Dst1
Src6-9-6-5-2-1-4-Dst2
Src6-9-6-5-8-7-Dst3
Src6-9-6-5-2-3-Dst4
Src6-9-6-Dst5
Src6-9-Dst6
```

Where, flows to destination Dst1 and Dst6 are audio flows, flows to destination Dst2 and Dst5 are CC flows, and flows to destination Dst3 and Dst4 are video flows. Each path carries 10 flows, e.g., the path "Src1-1-Dst1" carries 10 audio flows. It can be seen that the longest path contains 7 hops, and the bottleneck link involves link (2-3) and link (8-7), both of which have 10 audio flows, 60 video flows, and 10 CC flows.

According to the longest path and the expected E2E latency, the per-hop latency bound for each type of flow can be estimated, i.e., 700us for audio and CC flows, 1400us for video flows. This means that the deadline mechanism needs to provide appropriate delay levels, and the delay level mapped by audio and CC flows cannot be larger than 700us, and the delay level mapped by video flows cannot be larger than 1400us.

For simplicity, a unified delay resource pool is configured on each link in the network, although different links can indeed be configured differently. This unified delay resource pool must meet the resource allocation requirements on both bottleneck and non-bottleneck links, so we slightly increase the loading and assume that the number of each type of flows on a link reaches 60. Figure 18 shows a possible delay resource pool and the corresponding delay levels mapped by flows. Note that there are other possible resource pool designs as long as they meet schedulability conditions.

Delay Levels	Bursts (Kbits)	Bandwidth (Mbps)	Services Mapped
d1 (100 us)	b1 = 40	r1 = 10	
d2 (200 us)	b2 = 144	r2 = 30	CC
d3 (300 us)	b3 = 0	r3 = 0	
d4 (400 us)	b4 = 0	r4 = 0	
d5 (500 us)	b5 = 0	r5 = 0	
d6 (600 us)	b6 = 0	r6 = 0	
d7 (700 us)	b7 = 120	r7 = 96	Audio
d8 (800 us)	b8 = 0	r8 = 0	
d9 (900 us)	b9 = 0	r9 = 0	
d10 (1000 us)	b10 = 0	r10 = 0	
d11 (1100 us)	b11 = 720	r11 = 660	Video

Figure 18: Delay Resource Pool and Service Mapped

Where, the granularity of delay level is chosen at the level of 100 us based on the link capability (1 Gbps) and the concurrent bursts (984 Kbits) of three type of flows. Intuitively, if the link capability is larger, such as 10 Gbps, the granularity can be chosen to be smaller, such as at the level of 10us.

The maximum delay level d11 (1100 us) is selected according to the minimum regulated packet interval of any flow, i.e., d11 is not larger than any regulated packet interval of any flow. In this example, the regulated packet interval (i.e., packet_size / service_rate) for flows audio, video, and CC are 1.25 ms, 1.1 ms, and 5 ms, respectively.

All delay levels consume approximately 800 Mbps bandwidth due to slightly increasing the loading. 60 CC flows are mapped to delay level d2, 60 audio flows are mapped to delay level d7, and 60 video flows are mapped to delay level d11. The delay level d1 may be used for more urgent flows other than the three types of flows considered.

For example, on the bottleneck link (2-3), 10 audio flows will allocate <burst = 20 Kbits, bandwidth = 16 Mbps> from d7, 60 video flows will allocate <burst = 720 Kbits, bandwidth = 660 Mbps> from d11, and 10 CC flows will allocate <burst = 24 Kbits, bandwidth = 5 Mbps> from d2.

For example on the non-bottleneck link (8-9), 50 audio flows will allocate <burst = 100 Kbits, bandwidth = 80 Mbps> from d7, zero video flows will not allocate resources from d11, and 30 CC flows will allocate <burst = 72 Kbits, bandwidth = 15 Mbps> from d2.

If on-time mode is applied, each packet of the audio flow may experience per-hop latency 700 us, and each packet of the CC flow may experience per-hop latency 200 us, and each packet of the video flow may experience per-hop latency 1100 us.

If in-time mode is applied, the best per-hop latency experienced by a packet in any flow may be 0 (without considering intra-node forwarding delay F), and the theoretical worst-case latency may be the same as that in the on-time mode in the case of heavyweight loading. However, due to lightweight loading in this example, smaller worst-case latency can be achieved. For example, on the bottleneck link (2-3), the admitted burst aggregation is composed of CC 24 Kbits, audio 20 kbits, and video 720 Kbits in descending order of transmission priority, therefore, the worst-case per-hop latency experienced by the last packet of flows CC, audio, and video, is 24 us, 44 us, and 764 us, respectively, which are much smaller than the values in the on-time mode. Similarly, on the non-bottleneck link (8-9), the admitted burst aggregation is composed of CC 72 Kbits and audio 100 kbits, in descending order of transmission priority, therefore, the worst-case per-hop latency experienced by the last packet of flows CC and audio is 72 us and 172 us respectively, which are also much smaller than the values in the on-time mode.

NOTE:

* In the above process of resource allocation, the 10 flows carried on each path are individually allocated burst resources. This is the most general case, that is, although the 10 flows share the same path, they are assumed to be independent of each other. However, in some cases, if these 10 flows are treated as a macro flow and policing is executed at the network entrance node for the macro flow (the leaky bucket depth is still the maximum packet size, but the leaky bucket rate is the aggregation rate), and resources are reserved for the macro flow instead of the member flow, then less burst resources will be consumed and larger service scales can be supported.

* This example conforms to the scenarios described in Section 3.2.1 and Section 4.3.2 for the application of simplified schedulability condition where d_n is not larger than any regulated packet interval of any flow. Therefore, in Figure 18 there are remaining 76 Kbits bursts available for any other delay level d_i, to support more flows.

* Video flows have 30 back-to-back packets per single burst, and are being regulated on the flow entrance node, to support 60 video flows on each link. As discussed in Section 10, operators may increase the bucket depth for video flows to make the shaped pattern and the original arrival pattern as consistent as possible, but this will be harmful to service scale. There is a trade-off between burstiness, policing, and service scale.

13.3.2.2. Ring-Mesh Reference Topology

[I-D.ietf-detnet-dataplane-taxonomy] describes another hierarchical Ring-Mesh topology (Figure 19), where, node 1~9 are core routers, and each leaf group consists of 10 ring networks. Each ring network (Figure 20) has 8 nodes, with one node connected to the core by a separate inter-domain link.

The capacity of all the links in the core network is 10 Gbps. The capacity of all the links in the leaf network, including the inter-domain link, is 1 Gbps.

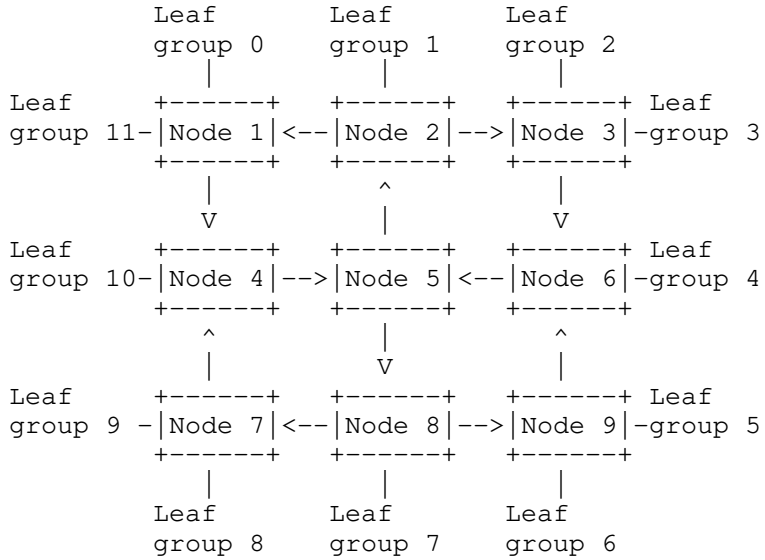

Figure 19: Hierarchical Ring-Mesh Topology

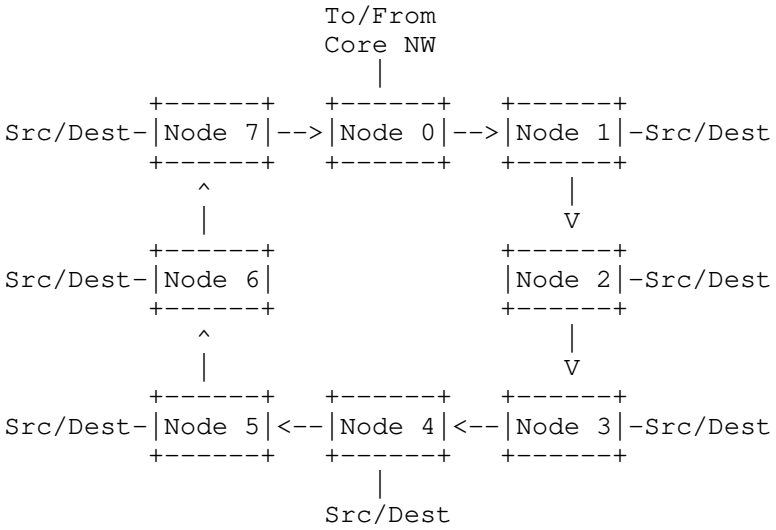

Figure 20: Ring Network

Again, three flow types, i.e., audio, video, and CC (Command and Control) are considered to require deterministic networking services, whose TSpec and RSpec are consistent with the above Grid example.

A flow-set is defined that includes 7 audio flows, 7 video flows, and 32 CC flows, all of which share the same DetNet path. For example, node 1 in the source ring may send a flow-set to node 7 in the destination ring, and the DetNet path may be, 1-2-3-4-5-6-7-0 (source ring), inter-domain link, core, inter-domain link, 0-1-2-3-4-5-6-7 (destination ring).

The longest DetNet path may be 20 hops, where, 7 hops in the source ring, 7 hops in the destination ring, 2 inter-domain hops, and 4 hops in the core.

The preset routing of DetNet path is that, every flow-set in a ring network travels from node i to node (i+7)mod8, and each leaf group i sends n flow-sets (e.g., n = 10, if a leaf group contains 10 ring networks) to the leaf group (i+6)mod12.

Take a flow-set from the source ring to the destination ring as the observed flow-set. The observed flow-set will compete with other 6 flow-sets in the ring, and compete with more flow-sets (coming from other leaf groups) in the core. Note that there is no competition on the inter-domain link.

In this example, we no longer assume that every packet of all flow-sets, including the observed flow-set and the competed flow-sets, arrives simultaneously. Although assuming extremely high concurrency can accommodate any topology with some actual concurrency, it underestimate the service scale that can be admitted. In fact, in the ring network, the concurrency at each hop is that only two input interfaces compete for one output interface. For inter-domain links, concurrency is even zero. In the core network, concurrency is also limited. By utilizing the knowledge of concurrency, more reasonable delay levels can be chosen to serve all flows.

In the ring network, on each hop, a bad flow interleaving is that there are two bursts competing for the outgoing interface. Their sizes are 1 flow-set and 6 flow-sets, respectively. The resolved size is 1 flow-set. Assign audio, video and CC to a single delay level d1. The resolved size of d1 is 174800 bits by 7 audio, 7 video, and 32 CC packets, introducing a maximum queueing delay of 174.8 us. The chosen d1 must not be less than 174.8 us. Considering that the transmission time of all bursts (i.e., all audio, video, and CC packets of 7 flow-sets) is 1.2236 ms, and the upcoming next round of 7 flow-sets will be the video packets after 1.1 ms and the audio packets after 1.25ms (with the resolved size 98000 bits and queueing delay 98 us), it can be seen that the transmission of current round of bursts will postpone the transmission of the upcoming next round of bursts, resulting in a postponement delay of 123.6 us (i.e., 1.2236 ms minus 1.1 ms), introducing a maximum queueing delay of 221.6 us (i.e., 123.6 us plus 98 us) for the next round of bursts. Similarly, walking through the following periodic rounds, it can be seen that the queuing delay will not exceed the above maximum value. Therefore, d1 (225 us) can be chosen for all flows in the ring network, with burst resource 1223600 bits and bandwidth 725 Mbps by 49 audio, 49 video, and 224 CC flows. Note that according to schedulability condition, the burst resource of d1 will affect the bounded delay of other delay levels with lower priority (if have).

For simplicity, a unified delay resource pool is configured on each link in the ring network, as shown in the following Figure 21.

Delay Levels	Bursts (Kbits)	Bandwidth (Mbps)	Services Mapped
d1 (225 us)	b1 = 1223.6	r1 = 725	CC/Audio/Video

Figure 21: Delay Resource Pool and Service Mapped in the Ring

In the core network, the details of all DetNet paths are as follows:

group0 -1-4-5-8-9- group6
group1 -2-1-4-5-8- group7
group2 -3-6-5-8-7- group8
group3 -3-6-5-8-7- group9
group4 -6-5-2-1-4- group10
group5 -9-6-5-2-1- group11
group6 -9-6-5-2-1- group0
group7 -8-7-4-5-2- group1
group8 -7-4-5-2-3- group2
group9 -7-4-5-2-3- group3
group10 -4-5-2-3-6- group4
group11 -1-4-5-8-9- group5

Where, the bottleneck link-4-5 will carry 70 flow-sets, in which, 10 flow-sets each from separate inter-domain link, 30 flow-sets from node 1, and 30 flow-sets from node 7.

Another bottleneck link-5-2 will also carry 70 flow-sets, in which, 30 flow-set from node 6, and 40 flow-sets from node 4.

On the bottleneck link-4-5, a bad flow interleaving is that there are 12 bursts competing for the outgoing interface. Their sizes are 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 30, and 30 flow-sets respectively. The resolved size is 40 flow-sets. Assign audio and CC to delay level d1 with high priority, and video to delay level d2 with low priority. The resolved size of d1 is 36320000 bits by 280 audio and 1280 CC packets, introducing a maximum queueing delay of 363.2 us. The resolved size of d2 is 69920000 bits by 280 audio, 1280 CC and 280 video packets, introducing a maximum queueing delay of 699.2 us. The chosen d1 and d2 must be larger than 363.2 and 699.2 us respectively. Considering that the transmission time of 12 incoming bursts (i.e., all audio, video, and CC packets of 70 flow-sets) is 1.2236 ms, and the upcoming next round of 70 flow-sets will be the video packets after 1.1 ms (with the resolved size 3360000 bits and queueing delay 336 us) and the audio packets after 1.25ms (with the resolved size 560000 bits and queueing delay 56 us), it can be seen that the transmission of current round of bursts will postpone the upcoming next round of bursts, resulting in a postponement delay of 123.6 us

(i.e., 1.2236 ms minus 1.1 ms), introducing a maximum queueing delay of 179.6 us (i.e., 123.6 us plus 56 us) for the next round of audio, and a maximum queueing delay of 557.6 us (i.e., 123.6 us plus 336 us, and plus 98 us by 490 audio packets) for the next round of video. Similarly, walking through the following periodic rounds, it can be seen that the queuing delay will not exceed the above maximum value. Therefore, in the core network, d1 (370 us) can be chosen for all audio and CC flows, with burst resource 6356000 bits and bandwidth 1860 Mbps by 490 audio and 2240 CC flows, and, d2 (700 us) can be chosen for all video flows, with burst resource 5880000 bits and bandwidth 5390 Mbps by 490 video flows.

The above chosen d1, d2 can also work for bottleneck link-5-2. For simplicity, a unified delay resource pool is configured on each link in the core network, with slightly increasing the loading and assuming that each link will carry 70 flow-sets, although different links can indeed be configured differently.

Figure 22 shows the delay resource pool and the corresponding delay levels mapped by flows in the core network.

Delay Levels	Bursts (Kbits)	Bandwidth (Mbps)	Services Mapped
d1 (400 us)	b1 = 6356	r1 = 1860	CC/Audio
d2 (700 us)	b2 = 5880	r2 = 5390	Video

Figure 22: Delay Resource Pool and Service Mapped in the Core

Other explanations are similar to the previous example of Grid reference topology. A noteworthy difference from the previous example is that the same flow is mapped to different delay levels between the ring domain and core domain to flexibly adapt to the large difference of service scale in these two domains.

14. IANA Considerations

There is no IANA requestion for this document.

15. Security Considerations

Security considerations for DetNet are described in detail in [RFC9055]. General security considerations for the DetNet architecture are described in [RFC8655]. Considerations specific to the DetNet data plane are summarized in [RFC8938].

Adequate admission control policies should be configured in the edge of the DetNet domain to control access to specific delay resources. Access to classification and mapping tables must be controlled to prevent misbehaviors, e.g., an unauthorized entity may modify the table to map traffic to an expensive delay resource, and competes and interferes with normal traffic.

16. Acknowledgements

TBD

17. References

17.1. Normative References

[I-D.hp-detnet-tsn-queuing-mechanisms-evaluation]
Yan, J., Han, Y., Peng, S., and Y. Gao, "Analysis and Evaluation for TSN Queuing Mechanisms", Work in Progress, Internet-Draft, draft-hp-detnet-tsn-queuing-mechanisms-evaluation-01, 20 December 2023, <https://datatracker.ietf.org/doc/html/draft-hp-detnet-tsn-queuing-mechanisms-evaluation-01>.

[I-D.ietf-detnet-dataplane-taxonomy]
Joung, J., Geng, X., Peng, S., and T. T. Eckert, "Dataplane Enhancement Taxonomy", Work in Progress, Internet-Draft, draft-ietf-detnet-dataplane-taxonomy-03, 2 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-dataplane-taxonomy-03>.

[I-D.ietf-detnet-scaling-requirements]
Liu, P., Li, Y., Eckert, T. T., Xiong, Q., Ryoo, J., zhushiyin, and X. Geng, "Requirements for Scaling Deterministic Networks", Work in Progress, Internet-Draft, draft-ietf-detnet-scaling-requirements-07, 20 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-scaling-requirements-07>.

[I-D.p-6man-deterministic-eh]
Peng, S., "Deterministic Source Route Header", Work in Progress, Internet-Draft, draft-p-6man-deterministic-eh-01, 10 October 2024, <https://datatracker.ietf.org/doc/html/draft-p-6man-deterministic-eh-01>.

[I-D.pb-6man-deterministic-crh]
Peng, S. and R. Bonica, "Deterministic Routing Header", Work in Progress, Internet-Draft, draft-pb-6man-

deterministic-crh-01, 10 October 2024, <https://datatracker.ietf.org/doc/html/draft-pb-6man-deterministic-crh-01>.

[I-D.peng-6man-deadline-option] Peng, S., Tan, B., and P. Liu, "Deadline Option", Work in Progress, Internet-Draft, draft-peng-6man-deadline-option-01, 11 July 2022, <https://datatracker.ietf.org/doc/html/draft-peng-6man-deadline-option-01>.

[I-D.peng-6man-delay-options] Peng, S., "Delay Options", Work in Progress, Internet-Draft, draft-peng-6man-delay-options-00, 18 January 2024, <https://datatracker.ietf.org/doc/html/draft-peng-6man-delay-options-00>.

[I-D.peng-detnet-policing-jitter-control] Peng, S., Liu, P., and K. Basu, "Mechanism to control jitter caused by policing in Detnet", Work in Progress, Internet-Draft, draft-peng-detnet-policing-jitter-control-01, 8 October 2024, <https://datatracker.ietf.org/doc/html/draft-peng-detnet-policing-jitter-control-01>.

[I-D.peng-lsr-deterministic-traffic-engineering] Peng, S., "IGP Extensions for Deterministic Traffic Engineering", Work in Progress, Internet-Draft, draft-peng-lsr-deterministic-traffic-engineering-03, 23 December 2024, <https://datatracker.ietf.org/doc/html/draft-peng-lsr-deterministic-traffic-engineering-03>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

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

[RFC2474] Nichols, K., Blake, S., Baker, F., and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, DOI 10.17487/RFC2474, December 1998, <https://www.rfc-editor.org/info/rfc2474>.

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[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>.

[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>.

[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/info/rfc9320>.

17.2. Informative References

[CQ-EDF] "Programmable Calendar Queues for High-speed Packet Scheduling", 2020, <https://dl.acm.org/doi/10.5555/3388242.3388292>.

[EDF-algorithm] "A framework for achieving inter-application isolation in multiprogrammed, hard real-time environments", 1996, <https://ieeexplore.ieee.org/document/896011>.

[EF-FIFO] "Fundamental Trade-Offs in Aggregate Packet Scheduling", 2001, <https://ieeexplore.ieee.org/document/992892>.

[IR-Theory]
"A Theory of Traffic Regulators for Deterministic Networks with Application to Interleaved Regulators", 2018, <https://ieeexplore.ieee.org/document/8519761>.

[Jitter-EDF]
"Delay Jitter Control for Real-Time Communication in a Packet Switching Network", 1991, <https://ieeexplore.ieee.org/document/152873>.

[MEDF] "MEDF - A Simple Scheduling Algorithm for Two Real-Time Transport Service Classes with Application in the UTRAN", 2003, <https://ieeexplore.ieee.org/document/1208948>.

[Net-Calculus]
"Network Calculus: A Theory of Deterministic Queuing Systems for the Internet", 2001, <https://leboudec.github.io/netcal/latex/netCalBook.pdf>.

[P802.1DC] "Quality of Service Provision by Network Systems", 2023, <https://1.ieee802.org/tsn/802-1dc/>.

[PIFO] "Programmable Packet Scheduling at Line Rate", 2016, <https://dl.acm.org/doi/pdf/10.1145/2934872.2934899>.

[RC-EDF] "Efficient Network QoS Provisioning Based on per Node Traffic Shaping", 1996, <https://ieeexplore.ieee.org/document/532860>.

[RC-EDF-para]
"Traffic Shaping for End-to-End Delay Guarantees with EDF Scheduling", 2000, <https://ieeexplore.ieee.org/document/847934>.

[RPQ-EDF] "Exact Admission Control for Networks with a Bounded Delay Service", 1996, <https://ieeexplore.ieee.org/document/556345/>.

[SCED] "SCED: A Generalized Scheduling Policy for Guaranteeing Quality-of-Service", 1999, <https://ieeexplore.ieee.org/document/803382>.

[SP-LATENCY]
"Guaranteed Latency with SP", 2020, <https://ieeexplore.ieee.org/document/9249224>.

Appendix A. Proof of Schedulability Condition for RPQ

Figure 23 below gives the proof of schedulability condition for RPQ.

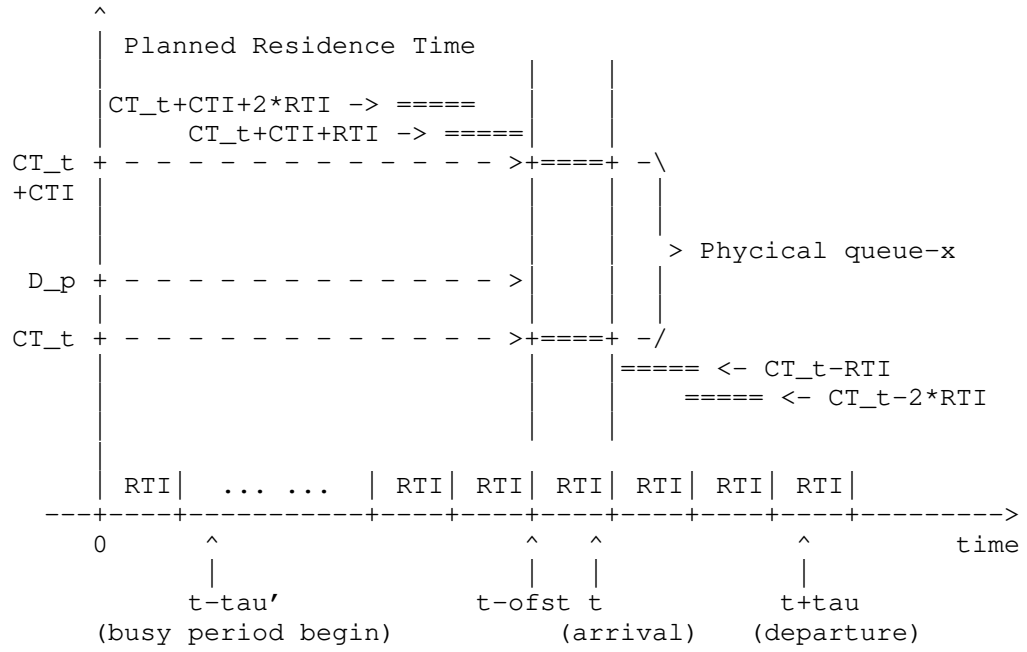

Figure 23: RPQ Based Scheduling

Suppose that the observed packet, with planned residence time D_p, arrives at the scheduler at time t and leaves the scheduler at time t+tau. It will be inserted to physical queue-x with count-down time CT_t at the current timer interval RTI with starting time t-ofst and end time t-ofst+RTI. According to the above packet queueing rules, we have CT_t <= D_p < CT_t+CTI. Also suppose that t-tau' is the beginning of the busy period closest to t. Then, we can get the amount of packets within time interval [t-tau', t+tau] that must be scheduled before the observed packet. In detailed:

* For all flow i with planned residence time D_i meeting CT_t <= D_i < CT_t+CTI, the workload is sum{A'_i[t-tau', t]}.

  Explanation: since the packets with planned residence time D_i in the range [CT_t, CT_t+CTI) arrived at time t will be sent before the observed packet, the packets with the same D_i before time t will become more urgent at time t, and must also be sent before the observed packet.

* For all flow i with planned residence time D_i meeting D_i >= CT_t+CTI, the workload is sum{A'_i[t-tau', t-ofst-(D_i-CT_t-CTI)]}.

  Explanation: although the packets with planned residence time D_i larger than CT_t+CTI arrived at time t will be sent after the observed packet, but the packets with the same D_i before time t, especially before time t-ofst-(D_i-CT_t-CTI), will become more urgent at time t, and must be sent before the observed packet.

* For all flow i with planned residence time D_i meeting D_i < CT_t, the workload is sum{A'_i[t-tau', t+(CT_t-D_i)]}.

  Explanation: the packets with planned residence time D_i less than CT_t at time t will certainly be sent before the observed packet, at a future time t+(CT_t-D_i) the packets with the same D_i will still be urgent than the observed packet (even the observed packet also become urgent), and must be sent before the observed packet.

* Then deduct the traffic that has been sent during the busy period, i.e., C*(tau+tau').

Let tau as D_p, and remember that CT_t <= D_p, the above workload is less than

```
sum{A'_i(tau'+CT_t+CTI-D_i) for all D_i >= CT_t} +
sum{A'_i(tau'+CT_t-D_i) for all D_i < CT_t} - C*(tau'+D_p)
```

It is further less than

```
sum{A'_i(tau'+D_p+CTI-D_i) for all D_i >= D_2} + A'_1(tau'+D_p-
D_1) - C*(tau'+D_p)
```

Then, denote x as tau'+D_p, we have

```
sum{A'_i(x+CTI-D_i) for all D_i >= D_2} + A'_1(x-D_1) - C*(x)
```

In the case that d_i contains only one D_i, we have A_i = A'_i, d_i = D_i, so the above workload is

```
sum{A_i(x+CTI-d_i) for all d_i >= d_2} + A_1(x-d_1) - C*(x)
```

Let the above workload be less than zero, then we get Equation-2.

In the case that d_i contains multiple D_i, e.g., d_1 is the minimum delay level with 10us, D_1 ~ D_10 is 10 ~ 19us respectively, d_2 is 20us, D_11 ~ D_20 iS 20 ~29us respectively, etc. Let D_1 ~ D_10 consume the resources of d_1, and D_11 ~ D_20 consume the resources of d_2, etc. Then, the above workload is less than

sum{A'_i(x+CTI-d_i) for all D_i belongs to d_i} - C*(x)

That is sum{A_i(x+CTI-d_i) for all d_i} - C*(x), and let it less than zero, then we get Equation-3.

Appendix B. Proof of Schedulability Condition for Alternate QAR of RPQ

In the case that d_i contains only one D_i, the schedulability condition is Equation-1. This is because, in the workload, for all D_i meeting D_i >= CT_t+CTI, their contributed workload is changed to sum{A'_i[t-tau', t-ofst-(D_i-CT_t)]} based on the analysis of Equation-2, that is, the amount of workload A'_i(CTI) (that is placed in queue-x) is excluded.

In the case that d_i contains multiple D_i, the schedulability condition is still Equation-3. This is because multiple D_i may belong to the same delay level as D_p. Assuming that within time zone [t-ofst, t-ofst+I] the list of all arrived D_i in the same parent queue-x with [CT_t, CT_t+CTI) as the observed packet (with D_p) is:

* D_a1~D_am, where D_a1 is closer to CT_t+CTI, they are larger than D_p (but smaller than CT_t+CTI) and belongs to a larger delay level than d_p (corresponding delay level of D_p).

* D_b1~D_bm, they are larger than D_p and belongs to the same delay level as d_p.

* D_p.

* D_c1~D_cm, they are smaller than D_p, and may belongs to the same delay level as d_p or a lower delay level than d_p.

So that both D_b1~D_bm and D_c1~D_cm should be scheduled before the observed packet. This is also true for these set of packets that have arrived in history.

Strictly, for D_a1, the contributed workload is sum{A'_i[t-tau', t-ofst+I-CTI]}, that is, only before time t-ofst+I-CTI the arrived packets of D_a1 will be placed in a more urgent queue-y with [CT_t, CT_t + CTI) than queue-x (at this history time its CT is [CT_t+CTI, CT_t + 2AT)) and should be shceduled before the observed packet.

Similarly, for D_a2, the contributed workload is sum{A'_i[t-tau', t-ofst+I-CTI+I]}, for D_am, the contributed workload is sum{A'_i[t-tau', t-ofst+I-CTI+(m-1)*I]}.

Note that queue-x also contains packets with D_i (e.g., D_a0, larger than D_a1) that have arrived in history. For D_a0, the contributed workload is sum{A'_i[t-tau', t-ofst+I-CTI-(D_a0-D_a1)]}.

However, the number of m is not fixed. For safety, we can appropriately overestimate workload time zone of D_a1~D_am to time instant t and regard that they need to be scheduled before the observed packet. Based on this, we can get the Equation-3.

Authors' Addresses

Shaofu Peng
ZTE Corporation
China
Email: peng.shaofu@zte.com.cn

Zongpeng Du
China Mobile
China
Email: duzongpeng@foxmail.com

Kashinath Basu
Oxford Brookes University
United Kingdom
Email: kbasu@brookes.ac.uk

Zuopin Cheng
New H3C Technologies
China
Email: czp@h3c.com

Dong Yang
Beijing Jiaotong University
China
Email: dyang@bjtu.edu.cn

Chang Liu
China Unicom
China

Email: liuc131@chinaunicom.cn

DetNet
Internet-Draft
Intended status: Standards Track
Expires: 10 October 2025

Shaofu. Peng
ZTE
Peng. Liu
China Mobile
Kashinath. Basu
Oxford Brookes University
Aihua. Liu
ZTE
Dong. Yang
Beijing Jiaotong University
Guoyu. Peng
Beijing University of Posts and Telecommunications

8 April 2025

Timeslot Queueing and Forwarding Mechanism
draft-peng-detnet-packet-timeslot-mechanism-11

Abstract

IP/MPLS networks use packet switching (with the feature store-and-forward) and are based on statistical multiplexing. Statistical multiplexing is essentially a variant of time division multiplexing, which refers to the asynchronous and dynamic allocation of link timeslot resources. In this case, the service flow does not occupy a fixed timeslot, and the length of the timeslot is not fixed, but depends on the size of the packet. Statistical multiplexing has certain challenges and complexity in meeting deterministic QoS, and its delay performance is dependent on the used queueing mechanism. This document further describes a generic time division multiplexing scheme for layer-3 in an IP/MPLS networks, which we call timeslot queueing and forwarding (TQF) mechanism. TQF is an enhancement based on TSN TAS and allows the data plane to create a flexible timeslot mapping scheme based on available timeslot resources. It defines a cyclic period consisting of multiple timeslots where a flow is assigned to be transmited within one or more dedicated timeslots. The objective of TQF is to better handle large scaling requirements.

Copyright Notice

Table of Contents

1. Introduction

IP/MPLS networks use packet switching (with the feature store-and-forward) and are based on statistical multiplexing. The discussion of supporting multiplexing in the network was first seen in the time division multiplexing (TDM), frequency division multiplexing (FDM) and other technologies of telephone communication network (using circuit switching). Statistical multiplexing is essentially a variant of time division multiplexing, which refers to the asynchronous and dynamic allocation of link resources. In this case, the service flow does not occupy a fixed timeslot, and the length of the timeslot is not fixed, but depends on the size of the packet. In contrast, synchronous time division multiplexing means that a sampling frame (or termed as time frame) includes a fixed number of fixed length timeslots, and the timeslot at a specific position is allocated to a specific service. The utilization rate of link resources in statistical multiplexing is higher than that in synchronous time division multiplexing. However, if we want to provide deterministic end-to-end delay in packet switched networks based on statistical multiplexing, the difficulty is greater than that in synchronous time division multiplexing. The main challenge is to obtain a deterministic upper bound on the queueing delay, which is closely related to the queueing mechanism used in the network.

In addition to IP/MPLS network, other packet switched network technologies, such as ATM, also discusses how to provide corresponding transmission quality guarantee for different service types. Before service communication, ATM needs to establish a connection to reserve virtual path/channel resources, and use fixed-length short cells and timeslots. The advantage of short cell is small interference delay, but the disadvantage is low encoding efficiency. The mapping relationship between ATM cells and timeslots is not fixed, so it still depends on a specific cells scheduling mechanism (such as [ATM-LATENCY]) to ensure delay performance.

Although the calculation of delay performance based on short and fixed-length cells is more concise than that of IP/MPLS networks based on variable length packets, they all essentially depend on the queueing mechanism.

[TAS] introduces a synchronous time-division multiplexing method based on gate control list (GCL) rotation in Ethernet LAN. Its basic idea is to calculate when the packets of the service flow arrive at a certain node, then the node will turn on the green light (i.e., the transmission state is set to OPEN) for the corresponding queue inserted by the service flow at that time duration, which is defined as TimeInterval between two adjacent items in gating cycle. The TimeInterval is exactly the timeslot resource that can be reserved for service flow. A set of queues is controlled by the GCL, with round robin per gating cycle. The gating cycle (e.g, 250 us) contains a lot of items, and each item is used to set the OPEN/CLOSED states of all traffic class queues. By strictly controlling the release time of service flow at the network entry node, multiple flows always arrive sequentially during each gating cycle at the intermediate node and are sent during their respective fixed timeslot to avoid conflicts, with extremely low queueing delay. However, the GCL state (i.e., items set, and different TimeInterval value between any two adjacent items) is related with all ordered flows that passes through the node. Calculating and installing GCL states separately on each node has scalability issues.

[CQF] introduces a synchronous time-division multiplexing method based on fixed-length cycle in Ethernet LAN. [ECQF] is a further enhancement of the classic CQF. CQF with 2-buffer mode or ECQF with x-bin mode only uses a small number of cycles to establish the cycle mapping between a port-pair of two adjacent nodes, which is independent of the individual service flow. The cycle mapping may be maintained on each node and swaped based on a single cycle id carried in the packet during forwarding ([I-D.eckert-detnet-tcqf]), or all cycle mappings are carried in the packet as a cycle stack and read per hop during forwarding ([I-D.chen-detnet-sr-based-bounded-latency]). According to [ECQF], how many cycles (i.e., x-bin mode) are required depends on the proportion of the variation in intra-node forwarding delay relative to the cycle size. If the proportion is small, 3-bin is enough, otherwise, more than 3 bins needed. Compared to TAS, CQF/ECQF no longer maintains GCL on each node, but instead replaces the large number of variable length of timeslots related to service flows in GCL with a small number of fixed length cycles unrelated to service flows. Thus, CQF/ECQF simplifies the data plane, but leaves the complexity to the control plane, by calculating and controlling the release time of service flow at the network entry, to guarantee no conflicts between flows in any cycle on any intermediate nodes.

In order to meet the large scaling requirements, this document describes a scheduling mechanism for enhancing TAS. It defines a cyclic period consisting of multiple timeslots that share limited buffer resources, and a flow is assigned to be transmited within one or more dedicated timeslots. It does not rely on time synchronization, but needs to detect and maintain the phase difference of cyclic period between adjacent nodes. It further defines two scheduling modes: on-time or in-time mode. We call this mechanism as Timeslot Queueing and Forwarding (TQF), as a supplement to IEEE 802.1 TSN TAS. In TQF, the selected length of the cyclic period (i.e., gating cycle of TAS) depends on the length of the supported service burst interval. Note that TQF is a layer-3 solution and can operate over different types of QoS sensitive layer-2 including TSN but is not an alternative to TSN.

Similar to TAS and CQF/ECQF, TQF is also TDM based scheduling mechanisms. However, their differences are as below:

* Compared to classic TAS, TQF may use round robin queues corresponding to the count of timeslots during gating cycle, while TAS only maintains queues corresponding to the number of traffic classes and one of them is used for the Scheduled Traffic (i.e., DetNet flows). That means TQF need more queues than TAS (i.e., multiple timeslot queues vs single traffic class queue). However, TAS needs to use other complex methods to control the arrival order of all flows sharing the same traffic class queue to isolate them (so that each flow faces almost zero queuing delay), while TQF's timeslot queue naturally isolates flows by timeslot id of gating cycle. And, TQF with in-time scheduling mode may use a single PIFO (put in first out) queue to approximate the ultra-low delay of TAS.

* Compared to CQF/ECQF, TQF on-time scheduling maintains round robin queues corresponding to the count of timeslots during gating cycle, while CQF/ECQF maintains extra tolerating queues depending on the proportion of the variation in intra-node forwarding delay relative to the cycle size. There is no gating cycle with its timeslot resources designed by CQF/ECQF, it needs to use additional flow interleaving method to control the arrival order of flows sharing the same cycle queue to isolate flows (or alternatively tolerate overprovision), while TQF's timeslot queue naturally isolates flows by timeslot id of gating cycle. This is also the semantic difference between cycle id and timeslot id, where the former is used to indicate the NO. of the aggregated queues such as sending, receiving, or tolerating queue, rather than indicating the individual timeslot resource within the gating cycle like the later. That is, after defining timeslot resources in IP/MPLS, TQF does not limit the implementations of the data structure type corresponding to timeslot resources on the data plane, which may be round robin queues, or a single PIFO queue.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

The following terminology is introduced in this document:

Timeslot: The unit of TQF scheduling. It needs to design a reasonable value, such as 10us, to send at least one complete packet. Different nodes can be configured with different length of timeslot.

Timeslot Scheduling: The packet is stored in the buffer zone corresponding to a specific timeslot id, and may be sent before (in-time mode) or within (on-time mode) that timeslot. The timeslot id is always a NO. from the orchestration period.

Service Burst Interval: The traffic specification of DetNet flow generally follows the principle of generating a specific burst amounts within a specific length of periodic burst interval. For example, a service generates 1000 bits of burst per 1 ms, where 1 ms is the service burs interval.

Orchestration Period: The orchestration period is a cyclic period

and used to instantiate timeslot resources on the link. The selection of orchestration period length depends on the service burst interval of DetNet flows, e.g., the Least Common Multiple of service burst intervals of all flows. It is actually the gating cycle in TAS, just with different queue allocation rules. It contains a fixed count (termed as N and numbered from 0 to N-1) of timeslots. For example, the orchestration period include 1000 timeslots and each timeslot length is 10 us. Multiple orchestration period length may be enabled on the link, but all nodes included in the DetNet path must interoperate based on the same orchestration period length. A specific orchestration period length can be used to indicate the corresponding TQF forwarding instance.

Ongoing Sending Period: The orchestration period which the ongoing sending timeslot belongs to.

Scheduling Period: The scheduling period of a TQF forwarding instance depends on the hardware's buffer resources that is supported by the device. Its length reflects the count of the timeslot resources (termed as M and numbered from 0 to M-1) with related buffer resources that is actually instantiated on the data plane, which is limited by hardware capabilities. Scheduling period length may be less than or equal to orchestration period length in the case of on-time mode, or larger than or equal to orchestration period length in the case of in-time mode. Packets belonging to a specific timeslot (in orchestration period) will be mapped and stored in the buffer zone of the corresponding timeslot of the scheduling period.

Incoming Timeslot: For the headend of the DetNet path, the current timeslot of UNI at which a flow arriving and after being policing is the incoming timeslot. For any intermediate node of the DetNet path, the timeslot contained in the packet received from the upstream node (i.e., the outgoing timeslot of the upstream node) is the incoming timeslot. An incoming timeslot id is the timeslot in the context of the orchestration period.

Outgoing Timeslot: When sending a packet to the outgoing port, according to resource reservation or certain rules, it chooses to send packet in the specified timeslot of that port, which is the outgoing timeslot. An outgoing timeslot id is the timeslot in the context of the orchestration period.

Ongoing Sending Timeslot: When the end of the incoming timeslot to

which the packet belongs reaches a specific port, the timeslot currently in the sending state is the ongoing sending timeslot of that port. Note that the ongoing sending timeslot is different with the outgoing timeslot. An ongoing sending timeslot id is the timeslot in the context of the orchestration period.

3. Overview

This scheme introduces the time-division multiplexing scheduling mechanism based on the fixed length timeslot in the IP/MPLS network. Note that the time-division multiplexing here is a L3 packet-level scheduling mechanism, rather than the TDM port (such as SONET/SDH) implemented in L1. The latter generally involves the time frame and the corresponding framing specification, which is not necessary in this document. The data structure associated with timeslot resources may be implemented using round robin queues, or a single PIFO queue, etc.

Figure 1 shows the TQF scheduling behavior implemented by the intermediate node P through which a deterministic path passes.

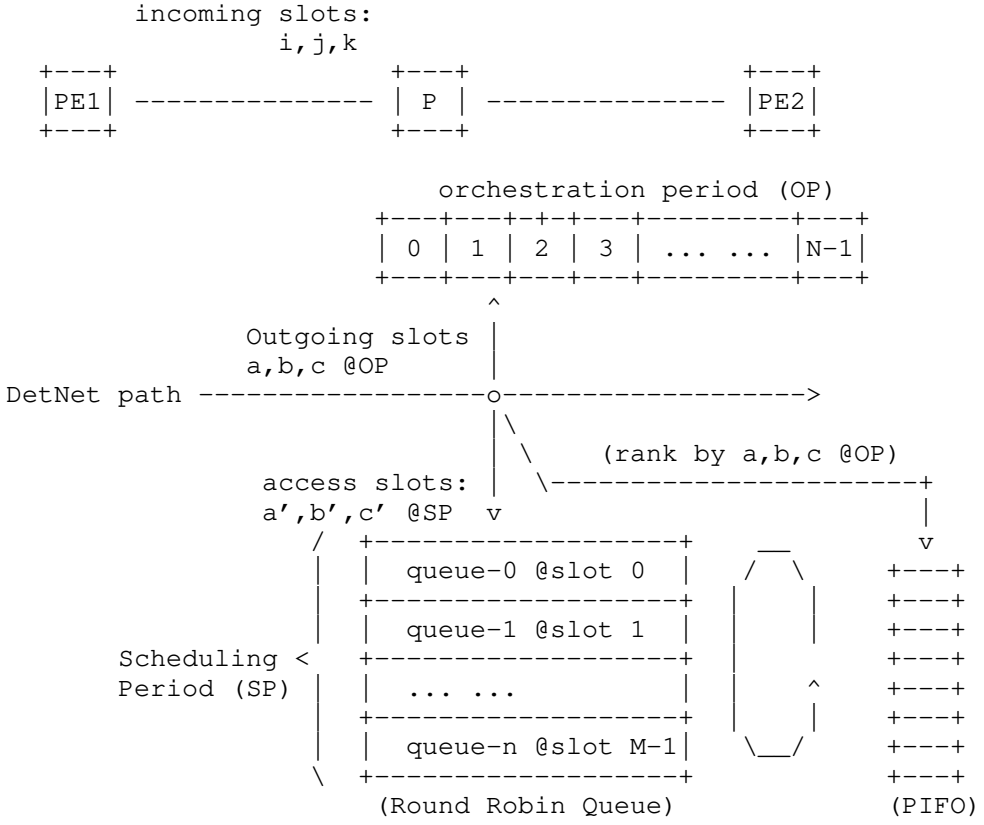

Figure 1: Timeslot Based Scheduling Mechanism

Where, both the orchestration period and the scheduling period consist of multiple timeslots, the number of timeslots supported by orchestration period is related to the length of the service burst interval, while the number of timeslots supported by scheduling period is limited by hardware capabilities, and it may be instantiated by a Round Robin queue or PIFO.

A TQF enabled link may configure multiple TQF scheduling instances each with specific orchestration period length. Nodes communicate with each other based on the same instance.

Each TQF scheduling instance related to a specific orchestration period length may configure its service rate, and the sum of service rates of all instances must not exceed the port bandwidth. For a TQF scheduling instance, the total amount of bits that can be consumed in each timeslot is generally not exceeding the result of the service rate multiplied by the timeslot length.

For each orchestration period length, all nodes in the network does not require phase alignment. The phase difference of orchestration period between adjacent nodes should be detected.

For a specific orchestration period configured on different links in the network, these links may configure different timeslot lengths because their capabilities vary, for example, the link capability of the edge nodes is weaker than that of core nodes.

In Figure 1, a DetNet path consumes timeslots i, j, k from the orchestration period of the link PE1-P, and a, b, c from the orchestration period of the link P-PE2 respectively. From node P's perspective, i, j, k are incoming timeslots, while a, b, c are outgoing timeslots. The cross connection between an incoming timeslot and an outgoing timeslot will result in the corresponding residence delay, which depends on the offset between the incoming and outgoing timeslots based on the phase difference of the orchestration periods of link PE1-P and P-PE2.

An outgoing timeslot in the orchestration period will finally access the mapped timeslot in the scheduling period. There is a mapping function from the timeslot z in the orchestration period to the timeslot z' in the scheduling period, i.e., z' = f(z). For example, the mapping function may be z' = z, z' = z + offset, z' = z % M, and z' = random(z), etc. Which function to use depends on the queue allocation rule and data structure instantiated for timeslot resources. In this document, we mainly discuss two mapping function, z' = z % M (in the case of round robin queue), and z' = z (in the case of PIFO).

How to calculate a DetNet path that meets the flow requirements is not within the scope of this document.

4. Timeslot Mapping Relationship

In order to determine the offset between the incoming timeslot and the outgoing timeslot in the context of specific TQF scheduling instance that is identified by orchestration period length, it is necessary to first determine the ongoing sending timeslot that the incoming timeslot falls into, i.e., the mapping relationship between the incoming timeslot and the ongoing sending timeslot.

Two methods are provided in the following sub-sections to determine the mapping relationship between the incoming timeslot and the ongoing sending timeslot.

4.1. Deduced by BTM

Figure 2 shows that there are three nodes U, V, and W in turn along the path. All nodes are configured with the same orchestration period length (termed as OPL), which is crucial for establishing a fixed timeslot mapping relationship between the adjacent nodes.

* Port_u2 has timeslot length L_u2, and an orchestration period contains N_u2 timeslots.

* Port_v1 has timeslot length L_v1, and an orchestration period contains N_v1 timeslots.

* Port_v2 has timeslot length L_v2, and an orchestration period contains N_v2 timeslots.

Hence, L_u2*N_u2 = L_v1*N_v1 = L_v2*N_v2. In general, the link bandwidth of edge nodes is small, and they will be configured with a larger timeslot length than the aggregated/backbone nodes.

It has been mathematically proven that if the least common multiple of L_u# and L_v# is LCM, OPL is also a multiple of LCM.

Node U may send a detection packet from the end (or head, the process is similar) of an arbitrary timeslot i of port_u2 connected to node V. After a certain link propagation delay (D_propagation), the packet is received by the incoming port of node V, and i is regarded as the incoming timeslot by V. At this time, the ongoing sending timeslot of port_v1 is j, and there is time T_ij left before the end of the timeslot j.

This mapping relationship is termed as:

* <instance OPL, port_u2 slot i, port_v1 slot j, T_ij>

To avoid confusion, we refer to this mapping relationship as the base timeslot mapping (BTM), as it is independent of the DetNet flows. Later, we will see the timeslot mapping relationship related to DetNet flow, which is the mapping relationship between the outgoing timeslot of port_u2 and the outgoing timeslot of port_v2, which is based on timeslot resource reservation and termed as the forwarding timeslot mapping (FTM).

BTM is generally maintained by node V when processing probe message received from node U. However, node U may also obtain this information from node V, e.g, by an ACK message. The advantage of maintaining BTM by node U is that it is consistent with the unidirectional link from node U to V, so it is more appropriate for node U (rather than V) to advertise it in the network. A BTM detection method can be found in [I-D.xp-ippm-detnet-stamp], and the advertisement method can be found in [I-D.peng-lsr-deterministic-traffic-engineering].

Note that this document does not recommend directly detecting and maintaining BTM between the outgoing timeslot of port_u2 and the ongoing sending timeslot of port_v2 (i.e., the outgoing port of downstream node V), as this is too trivial. In fact, as shown above, maintaining only BTM between the outgoing timeslot of port_u2 and the ongoing sending timeslot of port_v1 (i.e., the incoming port of downstream node V) is sufficient to derive other mapping relationships.

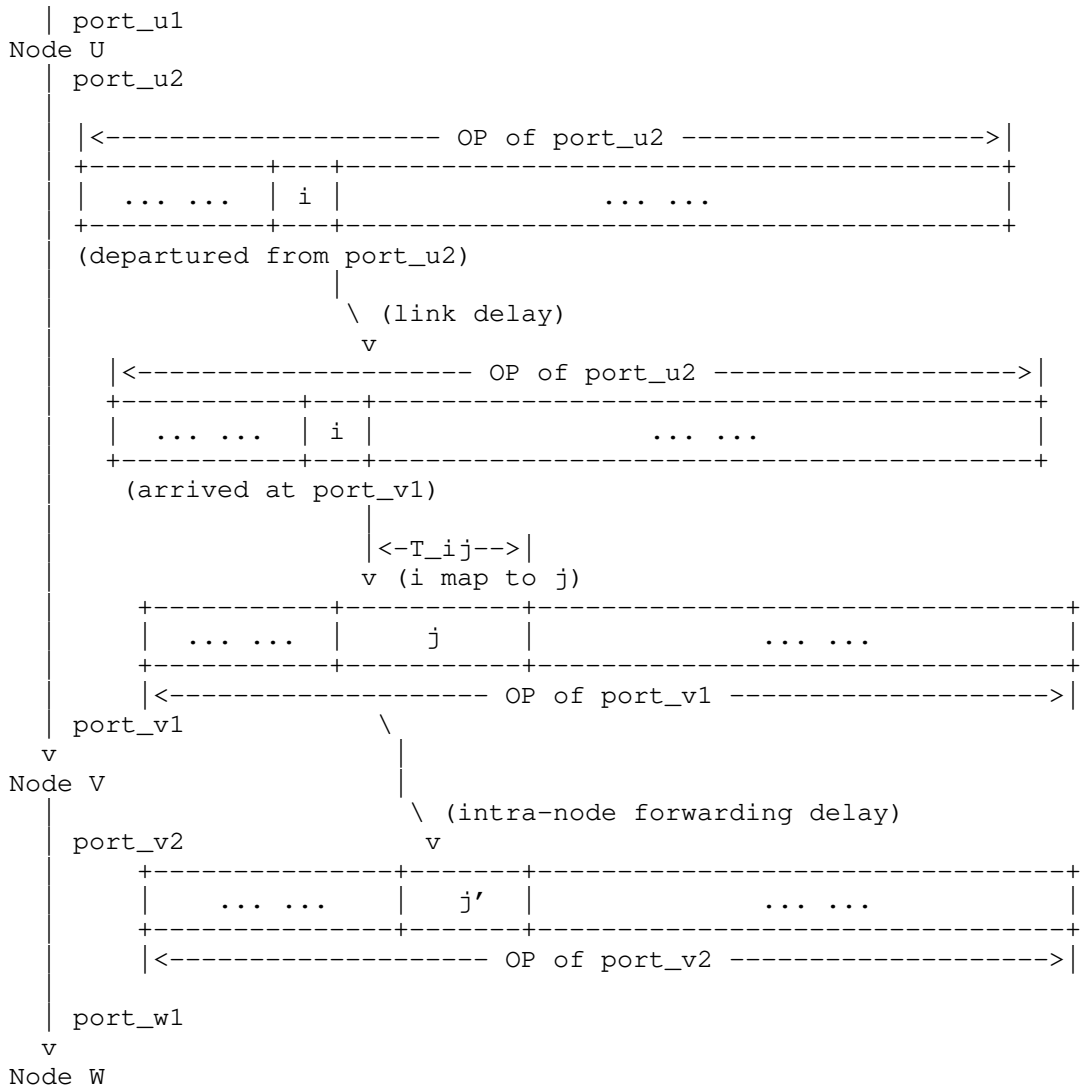

Figure 2: BTM Detection

Based on the above detected BTM, and knowing the intra-node forwarding delay (F) including parsing, table lookup, internal fabric exchange, we can derive BTM between any outgoing timelot x of port_u2 and the ongoing timeslot y of port_v2.

Let t is the offset between the end of the timeslot x of port_u2 and the beginning of the orchestration period of the port_v2.

* t = ((j'+1)*L_v1 - T_ij' + OPL + (x-i)*L_u2 + F) % OPL

Then,

* y = [t/L_v2]

And the time T_xy left before the end of the timeslot y is:

* T_xy = (y+1)*L_v2 - t

This document recommends that the time of each port within the same node must be synchronized, that is, all ports of a node share the same local system time, which is easy to achieve. It is also recommended that the begin time of the orchestration period for all ports within the same node be the same or differ by an integer multiple of OPL, e.g, maintaining a global initial time as the logical begin time for the first round of orchestration period for all ports. Whether node restart or port restart, this initial time should continue to take effect to avoid affecting the timeslot mapping relationship between each node. Depending on the implementation, considering that the initial time may be a historical time that is too far away from the current system time, regular updates may be made to it (e.g, self increasing k*OPL, where k is a natural number) to be closer to the current system time.

4.2. Deduced by BOM

Figure 3 shows that there are three nodes U, V, and W in turn along the path. Similar to Section 4.1, it still has L_u2*N_u2 = L_v1*N_v1 = L_v2*N_v2.

Node U may send a detection packet from the head (or end, the process is similar) of the orchestration period of port_u2 connected to node V. After a certain link propagation delay (D_propagation), the packet is received by the incoming port of node V. At this time, there is time P_uv left before the end of the ongoing sending period of port_v1.

This mapping relationship is termed as:

* <instance OPL, port_u2, port_v1, P_uv>

We refer to this mapping relationship as the base orchestration-period mapping (BOM), which it is independent of the DetNet flows.

BOM is generally maintained by node V when processing probe message received from node U. However, node U may also obtain this information from node V, e.g, by an ACK message. The advantage of

maintaining BOM by node U is that it is consistent with the unidirectional link from node U to V, so it is more appropriate for node U (rather than V) to advertise it in the network. A BOM detection method can be found in [I-D.xp-ippm-detnet-stamp], and the advertisement method can be found in [I-D.peng-lsr-deterministic-traffic-engineering].

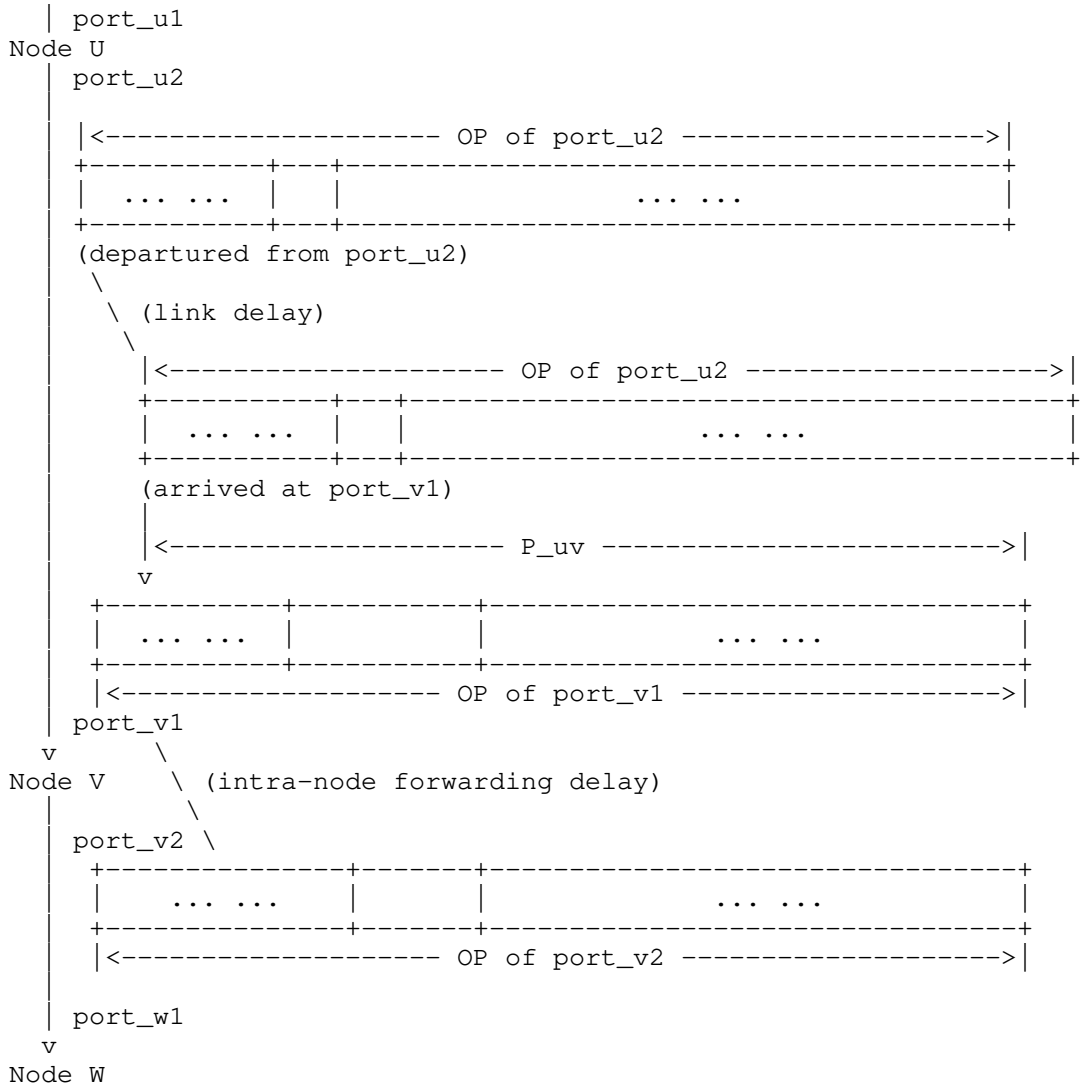

Figure 3: BOM Detection

Based on BOM, and knowing the intra-node forwarding delay (F), we can derive the mapping relationship between any outgoing timelot x of port_u2 and the ongoing timeslot y of port_v2.

Let t is the offset between the end of the timeslot x of port_u2 and the beginning of the orchestration period of the port_v2.

* t = ((x+1)*L_u2 + OPL - P_uv + F) % OPL

Then,

* y = [t/L_v2]

And the time T_xy left before the end of the timeslot y is:

* T_xy = (y+1)*L_v2 - t

5. Resources Used by TQF

The operation of TQF scheduling mechanism will consume two types of resources:

* Bandwidth: Each TQF scheduling instance may configure its service rate that is a dedicated bandwidth resource from the outgoing port, and the sum of service rates of all instances must not exceed the port bandwidth.

* Burst: The burst resources of a specific TQF scheduling instance can be represented as the corresponding bit amounts of all timeslots included in the orchestration period. The total amount of bits that can be consumed by flows in each timeslot is generally not exceeding the result of the service rate multiplied by the timeslot length.

6. Arrival Postion in the Orchestration Period

Generally, a DetNet flow has its TSpec, such as periodically generating traffic of a specific burst size within a specific length of burst interval, which regularly reaches the network entry. The headend executes traffic regulation (e.g, setting appropriate parameters for leaky bucket shaping), which generally make packets evenly distributed within the service burst interval, i.e, there are one or more shaped sub-burst in the service burst interval. There is an ideal positional relationship between the departure time (when each sub-burst leaves the regulator) and the orchestration period of UNI port, that is, each sub-burst corresponds to an ideal incoming timeslot of UNI port. Based on the ideal incoming timeslot, an ideal outgoing timeslot of NNI port may be selected and consumed by the

sub-burst.

For example, if a DetNet flow distributes m sub-bursts during the orchestration period, the network entry should maintain m items for that flow:

* <OPL, ideal incoming slot i_1, ideal outgoing slot z_1>

* <OPL, ideal incoming slot i_2, ideal outgoing slot z_2>

* ... ...

* <OPL, ideal incoming slot i_m, ideal outgoing slot z_m>

However, the packets arrived at the network entry are not always ideal, and the departure time from regulator may not be in a certain ideal incoming timeslot. Therefore, an important operation that needs to be performed by the network entry is to determine the ideal incoming timeslot i based on the actual departure time. This can first determine the actual incoming timeslot based on the actual departure time, and then match an item with the ideal incoming timeslot that is closest to and not earlier than the actual incoming timeslot. Alternatively, another match operation is to match an item with the ideal outgoing timeslot that is closest to and not earlier than the actual incoming timeslot. Note that all sub-bursts of the same flow must use the same match operation, otherwise, inconsistent operations may cause them to conflict in the same outgoing timeslot.

Figure 4 shows, for some typical DetNet flows, the ideal incoming timeslots in the orchestration period of UNI, as well as the ideal outgoing timeslots of NNI consumed by these DetNet flows.

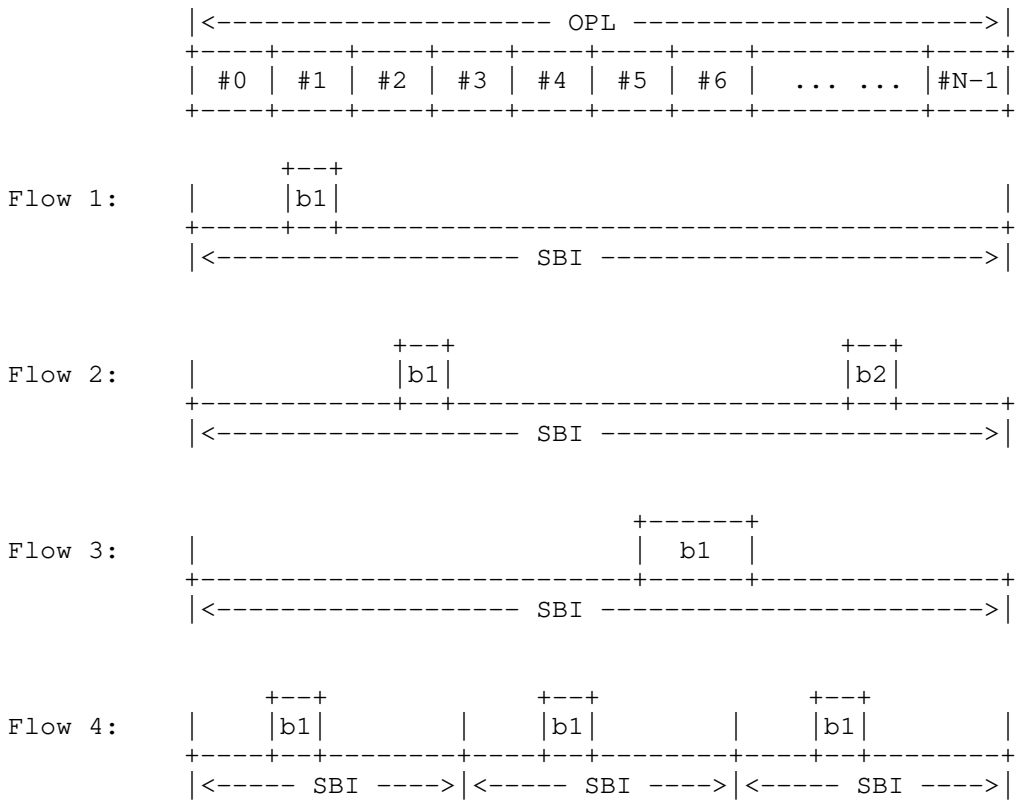

Figure 4: Relationship between SBI and OP

As shown in the figure, the service burst interval of flows 1, 2, 3 is equal to the orchestration period length, while the service burst interval of flow 4 is only 1/3 of the orchestration period length.

* Flow 1 generates a small single burst within its burst interval, which may consume timeslot 2 or other subsequent timeslot of NNI;

* Flow 2 generates two small discrete sub-bursts within its burst interval and also be shaped, which may respectively consume slots 4 and N-1 of NNI;

* Flow 3 generates a large single burst within its burst interval but not be really shaped (due to purchasing a larger burst resource and served by a larger bucket depth), which may also be split to multiple back-to-back sub-bursts and consume multiple consecutive timeslots, such as 8 and 9 of NNI.

* The service burst interval of flow 4 is only 1/3 of the orchestration period. Hence, construct flow 4' with 3 occurrence of the flow 4 within an orchestration period. So flow 4' is similar to flow 2, generating three separate sub-bursts within its burst interval. It may consume timeslots 3, 7, and N-1 of NNI.

7. Residence Delay Evaluation

7.1. Residence Delay on the Ingress Node

On the headend H, the received flow corresponds to an ideal incoming timeslot i of UNI port. Although there is actually no timeslot mapping relationship established between the end-system and the headend, we can still assume that the end-system applies the same orchestration period as UNI, and the BOM with phase aligned is detected. Then, according to Section 4, for the above incoming timeslot i, the ongoing sending timeslot j of NNI port, as well as the remaining time T_ij of timeslot j , can be deduced.

An outgoing timeslot z of NNI, which offset o (>=1) timeslots from j, can be selected for the flow. That is, z = (j+o)%N_h2, where N_h2 is the number of timeslots in the orchestration period of NNI.

Thus, on the headend H the residence delay obtained from the outgoing timeslot z is:

Best Residence Delay = F + T_ij + (o-1)*L_h2

Worst Residence Delay = F + T_ij + L_h1 + o*L_h2

Average Residence Delay = F + T_ij + (L_h1 + (2o-1)*L_h2)/2

where, L_h1 is the timeslot length of UNI port, L_h2 is the timeslot length of NNI port.

The best residence delay occurs when the flow arrived at the end of the ideal incoming timeslot i, and sent at the head of the outgoing timeslot z.

The worst residence delay occurs when the flow arrived at the head of the ideal incoming timeslot i, and sent at the end of the outgoing timeslot z.

The delay jitter within the headend is (L_h1 + L_h2). However, the jitter of the entire path is not the sum of the jitters of all nodes.

Note that there is a runtime jitter, as mentioned in Section 6, which depends on the deviation between the actual departure time t0 from the regulator and the begin time of the ideal incoming timeslot i, called the arrival deviation, i.e., i.begin - t0. A maximum arrival deivation can be set for each sub-burst of the flow. The maximum arrival deviation can be set as the interval between the adjacent ideal incoming timeslots. The scheduling period length also affect the value of maximum arrival deviation (refer to Section 13). Even if t0 is in the ideal incoming timeslot i, there will usually be a negative value of arrival deivation. Further operations to eliminate jitter (refer to Section 7.4) must consider any possible positions of t0 uniformly.

7.2. Residence Delay on the Transit Node

On the transit node V, according to Section 4, for any given incoming timeslot i, the ongoing sending timeslot j of the outgoing port (port_v2), as well as the remaining time T_ij of timeslot j, can be deduced.

An outgoing timeslot z of the outgoing port, which offset o (>=1) timeslots from j, can be selected for the flow. That is, z = (j+o)%N_v2, where N_v2 is the number of timeslots in the orchestration period of port_v2.

Thus, on the transit node V the residence delay obtained from the outgoing timeslot z is:

Best Residence Delay = F + T_ij + (o-1)*L_v2

Worst Residence Delay = F + T_ij + L_u2 + o*L_v2

Average Residence Delay = F + T_ij + (L_u2+(2o-1)*L_v2)/2

where, L_u2 and L_v2 is the timeslot length of port_u2 and port_v2 respectively.

The best residence delay occurs when the flow is received at the end of the incoming timeslot i and sent at the head of the outgoing timeslot z.

The worst residence delay occurs when the flow is received at the head of the incoming timeslot i and sent at the end of the outgoing timeslot z.

The delay jitter within the node is (L_u2 + L_v2). However, the jitter of the entire path is not the sum of the jitters of all nodes.

7.3. Residence Delay on the Egress Node

Generally, for the deterministic path carrying the DetNet flow, the flow needs to continue forwarding from the outgoing port of the egress node to the client side, and also faces the issues of queueing. However, the outgoing port facing the client side is the part of the overlay routing. It is possible to continue supporting TQF mechanism on that port. In this case, the underlay DetNet path will serve as a virtual link of the overlay path, providing a deterministic delay performance.

Therefore, for the underlay deterministic paths, the residence dalay on the egress node is only contributed by the forwarding delay (F) including parsing, table lookup, internal fabric exchange, etc.

7.4. End-to-end Delay and Jitter

Figure 5 shows that a path from headend P1 to endpoint E, for each node Pi, the timeslot length of the outgoing port is L_i, the intra-node forwarding delay is F_i, the remaining time of the mapped ongoing sending timeslot is T_i, the number of timeslots offset by the outgoing timeslot relative to the ongoing sending timeslot is o_i, especially on node P1 the timeslot length of UNI is L_h, then the end to end delay can be evaluted as follows (not including link propagation delay):

Best E2E Delay = sum(F_i+T_i+o_i*L_i, for 1<=i<=n) - L_n + F_e

Worst E2E Delay = sum(F_i+T_i+o_i*L_i, for 1<=i<=n) + L_h + F_e

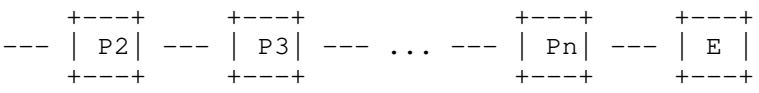

Figure 5: TQF Forwarding Path

The best E2E delay occurs when the flow arrived at the end of the ideal incoming timeslot and sent at the beginning of the outgoing timeslot of each node pi. The worst E2E delay occurs when the flow arrived at the beginning of the ideal incoming timeslot and sent at the end of the outgoing timeslot of each node Pi. The E2E delay jitter is (L_h + L_n).

Additional operation can be taken to further eliminate jitter. A latency deviation (E) can be included in the packet for this purpose. In order to let all packets of the same flow experience the same end-to-end delay that equals to the summary of the maximum arrival

deviation and the worst E2E delay, the flow entrance node can set E as maximum arrival deviation minus arrival deviation, and the flow exit node can update E as E plus departure deviation plus timeslot length, where departure deviation equals to the beginning of the outgoing timeslot minus the actual departure time from the scheduler. The departure deviation is also used to locate the right outgoing timeslot z in the case of in-time scheduling mode (refer to Section 10.2).

8. Flow States in Data-plane

The headend of the path needs to maintain the timeslot resource information with the granularity of sub-burst of each flow, so that each sub-burst of the DetNet flow can access the mapped timeslot resources. However, the intermediate node does not need to maintain states per flow, but only access the timeslot resources based on the timeslot id carried in the packets.

[I-D.pb-6man-deterministic-crh], [I-D.p-6man-deterministic-eh] defined methods to carry the stack of timeslot id in the IPv6 packets.

9. Queue Allocation Rule of Round Robin Queue

The buffer resources may be organized in the form of fixed number of round robin queues. In this case, only on-time scheduling mode is supported. In-time scheduling mode may cause urgent and non urgent packets to be stored in the same queue.

The number of round robin queues should be designed according to the number of timeslots included in the scheduling period. Each timeslot corresponds to a separate queue, in which the buffered packets must be able to be sent within a timeslot.

The length of the queue, i.e., the total number of bits that can be sent for a timeslot, equals to the allocated bandwidth of the corresponding TQF instance (see Section 12) multiplied by the timeslot length.

Figure 1 shows that the scheduling period actually instantiated on the data plane is not completely equivalent to the orchestration period. The scheduling period includes M timeslots (from 0 to M-1), while the orchestration period includes N timeslots (from 0 to N-1). N is an integer multiple of M. In the orchestration period, from timeslot 0 to M-1 is the first scheduling period, from timeslot M to slot 2M-1 is the second scheduling period, and so on. Therefore, it is necessary to convert the outgoing timeslot of the orchestration period to the target timeslot of the scheduling period, and insert the packet to the round robin queue corresponding to the target timeslot for transmission.

A simple conversion method is:

* target scheduling timeslot = outgoing timeslot % M

This is safe when o < M is always followed, where o is the number of offset timeslots between the outgoing timeslot z and the ongoing sending timeslot j (please refer to Section 7).

Next, we briefly demonstrate that the sub-burst that arrives at the outgoing port during the ongoing sending timeslot (j) can be safely inserted into the corresponding queue in the scheduling period mapped by the outgoing timeslot z, and that queue will not overflow.

Assuming that each timeslot in the orchestration period has a virtual queue, for example, termed the virtual queue corresponding to the outgoing timeslot z as queue-z, the packets that can be inserted into queue-z may only come from the following flows:

During the ongoing sending timeslot j = (z-M+1+N)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z) according to o = M-1.

During the ongoing sending timeslot j = (z-M+2+N)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z) according to o = M-2.

... ...

During the ongoing sending timeslot j = (z-1+N)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z) according to o = 1;

The total consumed burst resources of all these flows does not exceed the burst resource of the outgoing timeslot (z).

Then, when the ongoing sending timeslot changes to z, queue-z will be sent and cleared. In the following time, starting from timeslot z+1 to the last timeslot N-1, there are no longer any packets inserted into queue-z. Obviously, this virtual queue is a great waste of queue resources. In fact, queue-z can be reused by the subsequent outgoing timeslot (z+M)%N. Namely:

During the ongoing sending timeslot j = (z+1)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z+M)%N according to o = M-1.

During the ongoing sending timeslot j = (z+2)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z+M)%N according to o = M-2.

... ...

During the ongoing sending timeslot j = (z+M-1)%N, the flows that arrive at the outgoing port, that is, these flows may consume the outgoing timeslot (z+M)%N according to o = 1.

The total consumed burst resources of all these flows does not exceed the burst resource of the outgoing timeslot (z+M)%N.

It can be seen that queue-z can be used by any outgoing timeslot (z+k*M)%N, where k is a non negative integer. By observing (z+k*M)%N, it can be seen that the minimum z satisfies 0<= z< M, that is, the entire orchestration period actually only requires M queues to store packets, which are the queues corresponding to M timeslots in the scheduling period. That is to say, the minimum z is the timeslot id in the scheduling period, while the outgoing timeslot (z+k*M)% N is the timeslot id in the orchestration period. The latter obtains the former by moduling M, which can then access the queue corresponding to the former. In short, the reason why a queue can store packets from multiple outgoing timeslots without being overflowed is that the packets stored in the queue earlier (more than M timeslots ago) have already been sent.

For example, if the total length of all queues supported by the hardware is 4G bytes, the queue length corresponding to a timeslot of 10us at a port rate of 100G bps is 1M bits, then a maximum of 32K timeslot queues can be provided, and TQF scheduler can use some of the queue resources, e.g., M may be 1K queues to construct a scheduling periold with 10 ms, and the corresponding orchestration period may be several 10ms.

10. Queue Allocation Rule of PIFO Queue

The buffer resources may also be organized in the form of PIFO queue. In this case, both in-time and on-time scheduling mode can be easily supported, because packets with different rank always ensure scheduling order.

10.1. PIFO with On-time Scheduling Mode

In the case of on-time mode, the buffer cost of PIFO queue is the same as that of round robin queues. It can directly use the begin time of the outgoing timeslot z as the rank of the packet and insert the packet into the PIFO for transmission.

* rank = z.begin

Here, the outgoing timeslot z refers to the outgoing timeslot z that is after the arrival time at the scheduler and closest to the arrival time.

The rule of the on-time scheduling mode is that if the PIFO is not empty and the rank of the head of queue is equal to or earlier than the current system time, the head of queue will be sent; otherwise, not.

10.2. PIFO with In-time Scheduling Mode

In the case of in-time mode, the buffer cost of PIFO queue is generally larger than that of on-time mode due to burst accumulation. [SP-LATENCY] provides guidance for evaluating excess buffer requirements.

Similar to Section 10.1, it can directly use the begin time of the outgoing timeslot z as the rank of the packet and insert the packet into the PIFO for transmission. However, due to in-time scheduling behavior, the expected outgoing timeslot z may not be the current outgoing timeslot z that is closest to and after the arrival time at the scheduler, since there are repeated z in different rounds of orchestration period. The expected outgoing timeslot z may be far away from the arrival time.

A departure deviation may be carried in the packet to help locate the expected outgoing timeslot z.

On the headend node:

* Match the ideal incoming timeslot i and outgoing timeslot z based on the departure time from the regulator.

* rank = z.begin

* When the packet leaves the headend, the departure deviation is set to z.begin minus the actual departure time from the scheduler. The departure deviation is carried in the sending packet.

On the intermediate node:

* Obtain departure deviation from the received packet.

* Use the result of the arrival time plus the departure deviation as the ideal arrival time, to locate the expected outgoing timeslot z that is after and closest to this result.

* rank = z.begin

* When the packet leaves the intermediated node, the departure deviation is updated to z.begin minus the actual departure time from the scheduler. The updated departure deviation is carried in the sending packet.

The rule of the in-time scheduling mode is that as long as the PIFO is not empty, packets are always obtained from the head of queue for transmission.

In summary, the in-time scheduling with the help of departure deviation, can suffer from the uncertainty caused by burst accumulation, and it is recommended only deployed in small networks, i.e., a limited domain with a small number of hops, where the burst accumulation issue is not serious; The on-time scheduling is recommended to be used in large networks.

11. Global Timeslot ID

The outgoing timeslots discussed in the previous sections are local timeslots style for all nodes. This section discusses the situation based on global timeslot style.

Global timeslot style refers to that all nodes in the path are identified with the same timeslot id, which of course requires all nodes to use the same timeslot length. There is no need to establish FTM for the DetNet flow on each node or carry FTM in packets. The packet only needs to carry the unique global timeslot id. However, the disadvantage is that the latency performance of the path may be large, which depends on BOM between the adjacent nodes. Another disadvantage is that the success rate of finding a path that matches the service requirements is not as high as local timeslot style.

Global timeslot style requires that the orchestration period is equal to the scheduling period, mainly considering that arrival packets with any global timeslot id can be successfully inserted into the corresponding queue without overflow. However, as the ideal design goal is to keep the scheduling period less than the orchestration period, further research is needed on other methods (such as basically aligning orchestration period between nodes), to ensure that packets with any global timeslot id can queue normally when the scheduling period is less than the orchestration period.

Compared to the local timeslot style, global timeslot style means that the incoming timeslot i must map to the outgoing timeslot i too.

As the example shown in Figure 6, each orchestration period contains 6 timeslots. Node V has three connected upstream nodes U1, U2, and U3. During each hop forwarding, the packet accesses the outgoing timeslot corresponding to the global timeslot id and forwards to the downstream node with the global timeslot id unchanged. For example, U1 sends some packets with global slot-id 0, termed as g0, in the outgoing timeslot 0. The packets with other global slot-id 1~5 are similarly termed as g1~g5 respectively. The figure shows the scheduling results of these 6 batches of packets sent by upstream nodes when node V continues to send them.

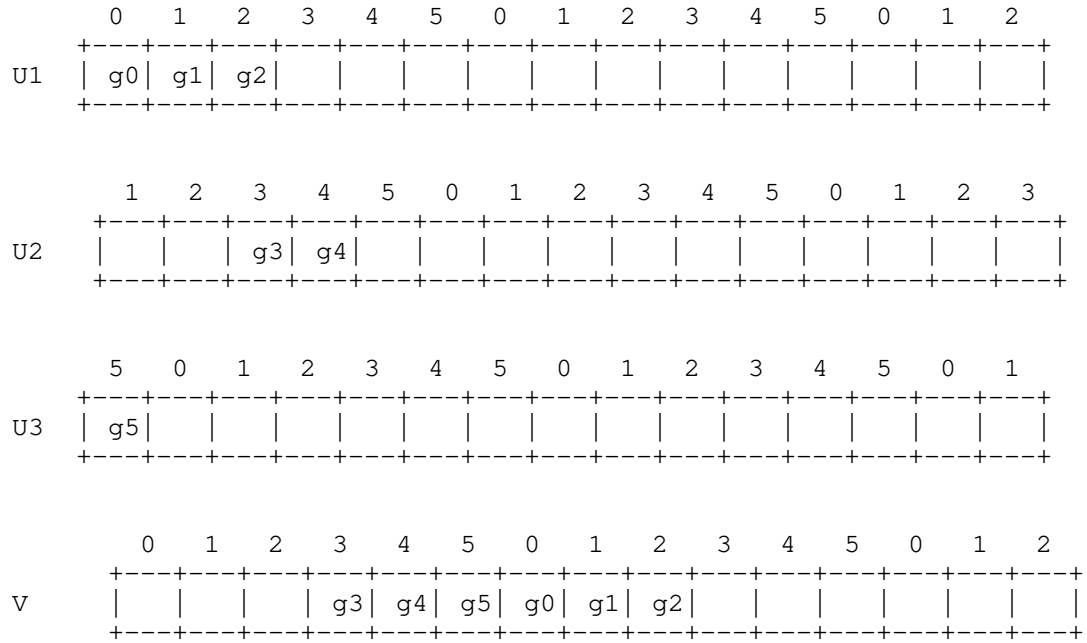

Figure 6: Global Timeslot Style Example

In this example:

* BTM between the outgoing timeslot of U1 and the ongoing sending timeslot of V is i -> i, so the global outgoing timeslot for the incoming timeslot i is i+6 (i.e., belongs to next round of orchestration periold).

* BTM between the outgoing timeslot of U2 and the ongoing sending timeslot of V is i -> i-1, so the global outgoing timeslot for the incoming timeslot i is i (i.e., belongs to current round of orchestration periold).

* BTM between the outgoing timeslot from U3 and the ongoing sending timeslot of V is i -> i+1, so the global outgoing timeslot for the incoming timeslot i is i+6 (i.e., belongs to next round of orchestration periold).

It can be seen that packets from U1 and U3 has large residency delay in the node V, while packets from U2 has small residency delay in the node V.

It should be noted that if round robin queue is used, for the orginal BTM i -> i (example of U1), or i -> i+1 (example of U3), the packets need to be stored in a buffer prior to the TQF scheduler (such as the buffer on the input port side) for a fixed latency (such as serveral timeslots) and then released to the scheduler. Otherwise, directly inserting the queue may cause jitter, i.e., part of the packets belonging to the same incoming timeslot i can be sent in the outgoing timeslot i, while the other part of the packets has to be delayed to be sent in the next round of timeslot i. This fixed-latency buffer is only introduced for specific upstream nodes. It can be determined according to the initial detection result of BTM between the adjacent nodes. If the original BTM is i -> i or i -> i+1, it needed, otherwise not. After the introduction of fixed-latency buffer, the new detection result of BTM will no longer be i -> i, or i -> i+1.

If PIFO queue is used, there is no need to introduce a fixed-latency buffer because in this case, rank = i.begin + OPL, and it will not be scheduled to be sent in the current outgoing timeslot i, but in the next round. However, in this case the PIFO itself serves as a fixed-latency buffer.

For the headend, the residence delay is similar to Section 7.1. For a flow which has the ideal incoming timeslot i, it may select a global outgoing timeslot z based on the BTM i -> j, where, j is the ongoing sending timeslot of the ougtoing port, and the timeslot offset o equals (N+z-j)%N.

For transit nodes, the residence delay is similar to Section 7.2, in addition to considering possible latency contributed by the above fixed-latency buffer. For a flow with the global incoming timeslot z, it still select the global outgoing timeslot z based on the BTM z -> j, where, j is the ongoing sending timeslot of the ougtoing port, and the timeslot offset o equals (N+z-j)%N.

The end-to-end delay equation is similar to Section 7.4, in addition to considering possible cumulated latency contributed by the above fixed-latency buffer.

12. Multiple Orchestration Periods

A single orchestration period may not be able to cover a wide range of service needs, such as some with a burst interval of microseconds, while others have a burst interval of minutes or even larger. When using a single orchestration period to simultaneously serve these services, the timeslot length must be microseconds, but the orchestration period length is minutes or more, resulting in the need to include a large number of timeslots in the orchestration period. The final result is a proportional increase in the buffer size required for the scheduling period.

Multiple orchestration periods each with different length may be provided by the network. A TQF enabled link can be configured with multiple TQF scheduling instances each corresponding to specific orchestration period length. For simplicity, the orchestration period length itself can be used to identify a specific instance.

For example, one orchestration period length is 300 us, termed as OPL-300us, which is the LCM of the burst interval of the set of flows served. Another orchestration period length is 100 ms, termed as OPL-100ms, which is the LCM of the burst interval of another set of flows served. Each orchestration period instance has its own timeslot length. The timeslot length of a long orchestration period instance should be longer than that of a short orchestration period instance, and the former is an integer multiple of the latter. But the long orchestration period itself may not necessarily be an integer multiple of the short orchestration period.

As shown in Figure 7, both link-a and link-b are configured with n orchestration period instances, with the corresponding orchestration period lengths OPL_1, OPL_2, ..., OPL_n in ascending order. For each orchestration period length OPL_i, the dedicated bandwidth resource is BW_U_i for node U (or BW_V_i for node V), and the timeslot length is TL_U_i for node U (or TL_V_i for node V). For each TQF enabled link, the sum of dedicated bandwidth resources of all TQF scheduling instances must not exceed the total bandwidth of the link.

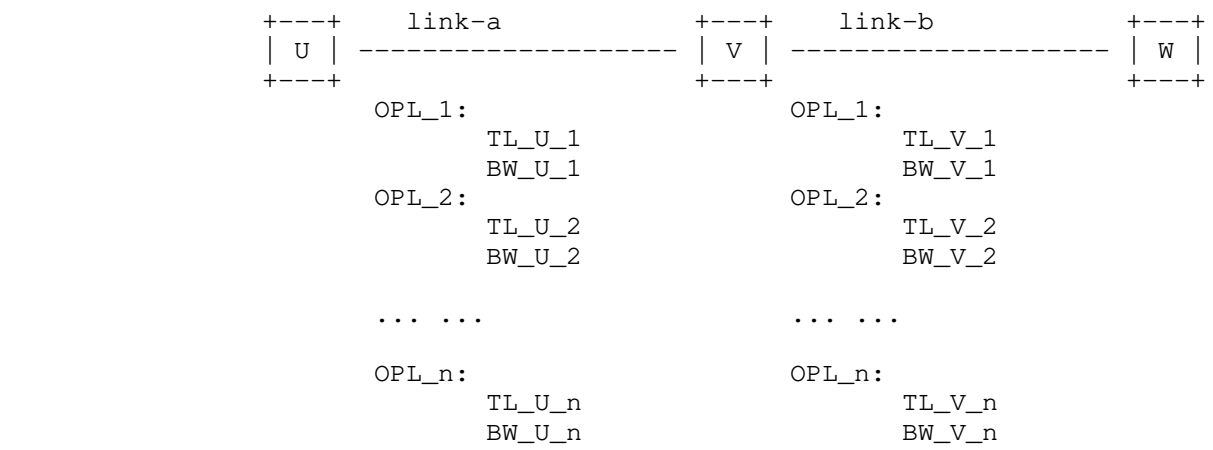

Figure 7: Multiple TQF Instances

Due to the fact that long orchestration periods serve DetNet flows with large burst intervals, for a given burst size, the larger the burst interval, the less bandwidth consumed by the DetNet flow. Therefore, it is recommended that the bandwidth resources of the long orchestration period is less than that of the short orchestration period, which is beneficial for reducing the buffer required for long orchestration period.

Interworking between different nodes is based on the same orchestration period. That means that the timeslot mapping described in Section 4 should be maintained in the context of the specific orchestration period. The orchestration period length should be carried in the forwarding packets to let the DetNet flow to consume the timeslot resources corresponding to the TQF scheduling instance.

If round robin queues are used, each TQF scheduling instance has its own separate queue set. Time division multiplexing scheduling is based on the granularity of the minimum timeslot length of all instances. Within each time unit of this granularity, the queues in the sending state of all instances are always scheduled in the order of OPL_1, OPL_2, ..., OPL_n.

If PIFO queue is used, all TQF scheduling instances may share a single PIFO queue. An implementation may use rank (i.e., the beginning of the outgoing timeslot z) plus timeslot length to determine the insertion order of two packets from different instances, so that the packet from the short orchstration period inserted at the front.

13. Admission Control on the Headend

On the network entry, traffic regulation must be performed on the incoming port, so that the DetNet flow does not exceed its T-SPEC such as burst interval, burst size, maximum packet size, etc. This kind of regulation is usually the shaping using leaky bucket combined with the incoming queue that receives DetNet flow. A DetNet flow may contain discrete multiple sub-bursts within its periodic burst interval. The leaky bucket depth should be larger than the maximum packet size, and should be consistent with the reserved burst resources required for the maximum sub-burst.

The scheduling mechanism described in this document has a requirement on the arrival time of DetNet flows on the network entry. It is expected that the distribution of sub-bursts (after regulation) of the DetNet flow will always appear in an ideal incoming timeslot of UNI port. However, a maximum arrival deviation is permitted. Based on the ideal incoming timeslot, an ideal outgoing timeslot is selected. For a single DetNet flow, the network entry may maintain multiple forwarding states each containing <ideal incoming timeslot, ideal outgoing timeslot>, due to many sub-bursts within the service burst interval. It is possible to set different maximum arrival deviation for different sub-bursts of the same flow, to eliminate the jitter between them.

For example, the network entry may maintain up to 3 sub-burst forwarding states for a flow. Ideally, all packets of this flow are split into 3 sub-bursts after regulation, each sub-burst matching one of the states. Here, 3 is the maximum sub-bursts for this flow, and it does not always contain so many bursts within the burst interval during actual sending.

The arrival deviation should not exceed o-1 for late arrival case, or M-o-1 for early arrival case, where o is the offset between the outgoing timeslot and ongoing sending timeslot as mentioned above. Intuitively, large o can tolerate large late arrival deviations, while small o (or large M even for large o) can tolerate large early arrival deviations.

Restricted arrival deviation is beneficial for the design goal that scheduling period is smaller than the orchestration period, and packets can always be successfully inserted into the scheduling queue (RR or PIFO) without overflow. An unrestricted arrival deviation may contain one or more scheduling periods, and therefore there is an overflow risk when inserting packets into the queue associated with the ideal outgoing timeslot z at the departue time from the regulator.

Otherwise, for randomly arriving DetNet flows, it can be supported by taking a large M (or even M = N) (option-1) to accommodate random arrival, or it can be supported by introducing an explicit buffer put before the scheduler on the network entry to let the arrival time always meet the arrival deviation limitation (option-2).

* Note that due to randomness of arrival time, the packet may just miss the scheduling (or arrive too earlier) and need to wait in the scheduling queue (in the case of option-1) or the explicit buffer (in the case of option-2) for the next orchestration period.

Note that the arrival deviation on the ingress node, also as the departure deviation on the egress node, may cause runtime jitter during forwarding. A latency deviation (E) calculated by arrival deviation and departure deviation can be used to eliminate jitter at the network egress on demand. This will achieve zero jitter performance metrics.

14. Frequency Synchronization

The basic explanation for frequency synchronization is that the crystal frequency of the hardware is consistent, which enables all nodes in the network to be in the same inertial frame and have the same time lapse rate. This is a prerequisite for TQF mechanism. The related frequency synchronization mechanisms, such as IEEE 1588-2008 Precision Time Protocol (PTP) [IEEE-1588] and synchronous Ethernet (syncE) [syncE], are not within the scope of this document.

Sometimes, people also refer to the frequency asynchrony as the timeslot rotation frequency difference caused by different node configurations with different timeslot lengths. This document supports the interconnection between nodes with this type of frequency asynchrony.

15. Evaluations

15.1. Large Scaling Requirements Matching Degree

This section gives the evaluation results of the TQF mechanism based on the requirements that is defined in [I-D.ietf-detnet-scaling-requirements].

Requirements	Evaluation	Notes
3.1 Tolerate Time Asynchrony	Yes	No time sync needed, only need frequency sync (3.1.3).
3.2 Support Large Single-hop Propagation Latency	Yes	The timeslot mapping covers any value of link propagation delay.
3.3 Accommodate the Higher Link Speed	Partial	The higher the service rate, the more buffer needed for the same timeslot length.
3.4 Be Scalable to the Large Number of Flows and Tolerate High Utilization	Yes	Multiple OPL instance, each for a set of serivce flows, without overprovision. Utilization may reach 100% link bandwidth. The unused bandwidth of the timeslot can be used by best-effot flows. Calculating paths is NP-hard.
3.5 Tolerate Failures of Links or Nodes and Topology Changes	N/A	Independent of queueing mechanism.
3.6 Prevent Flow Fluctuation	Yes	Flows are isolated from each other through timeslots.
3.7 Be scalable to a Large Number of Hops with Complex Topology	Yes	E2E latency is liner with hops , from ultra-low to low latency by multiple OPL. E2E jitter is low by on-time mode. Calculating paths is NP-hard.
3.8 Support Multi-Mechanisms in Single Domain and Multi-Domains	N/A	Independent of queueing mechanism.

Figure 8: Evaluation for Large Scaling Requirements

15.2. Taxonomy Considerations

[I-D.ietf-detnet-dataplane-taxonomy] provides criteria for classifying data plane solutions.

For performance, the per hop latency dominant factor of TQF is the offset between incoming timeslot and outgoing timeslot that is asigned to the flow.

For functional characteristics, TQF is periodic, flow level for traffic granularity, and bouned for time bounds.

* Periodic: Periodicity of TQF contains two characteristics, the first is that there is a time period P (i.e., orchestration periold) containing multiple time slots, and the second is that a flow is assigned repeatly to a particular set of time slots in that time period P.

* Flow level: TQF enhances TAS by extending the number of queues for Scheduling Traffic (i.e., from 1 to N), and may allocate one queue (represented by timeslot id) to each flow or a flow aggregate. Customizing different timeslots for different flows will make them interleaved.

* Bounded: The transmission completion of a packet will be in the outgoing timeslot to which the packet belongs, i.e., after the beginning of the timeslot and before the end of the timeslot.

[I-D.ietf-detnet-dataplane-taxonomy] also specifies the suitable categories of solutions for DetNet. According to the above functional characteristics, TQF will map to flow level periodic bounded category.

15.3. Examples

This section describes the example of how the TQF mechanism supports DetNet flows with different latency requirements.

15.3.1. Heavyweight Loading Example

This example observes the service scale and different latency bound supported by the TQF mechanism in the heavyweight loading case.

Figure 9 provides a typical reference topology that serves to represent or measure the multihop jitter and latency experience of a single "flow i" across N hops (in the figure, N=10). On each of the N outgoing interfaces (represented by circles in the figure), "flow i" has to compete with different flows (represented by different

symbols on each hop). Especially, the competed flows arrive simultaneously at multiple incoming ports, with the same starting time when measuring their respective residence time. The characteristic of this reference topology is that every link that "flow i" passes through may be a bottleneck link with 100% network utilization, causing "flow i" to achieve the worst-case latency on each hop.

As shown in Figure 9:

* Network transmission capacity: each link has rate 10 Gbps. Assuming the service rate of TQF scheduler allocate the total port bandwidth.

* TSpec of each flow, maybe:

  - burst size 1000 bits, SBI 1 ms, and average arrival rate 1 Mbps.

  - or, burst size 1000 bits, SBI 100 us, and average arrival rate 10 Mbps.

  - or, burst size 1000 bits, SBI 10 us, and average arrival rate 100 Mbps.

  - or, burst size 10000 bits, SBI 10 ms, and average arrival rate 1 Mbps.

  - or, burst size 10000 bits, SBI 1 ms, and average arrival rate 10 Mbps.

  - or, burst size 10000 bits, SBI 100 us, and average arrival rate 100 Mbps.

* RSpec of each flow, maybe:

  - E2E latency 100us, and E2E jitter less than 10us or 100us.

  - or, E2E latency 200us, and E2E jitter less than 20us or 200us.

  - or, E2E latency 300us, and E2E jitter less than 30us or 300us.

  - or, E2E latency 400us, and E2E jitter less than 40us or 400us.

  - or, E2E latency 500us, and E2E jitter less than 50us or 500us.

  - or, E2E latency 600us, and E2E jitter less than 60us or 600us.

- or, E2E latency 700us, and E2E jitter less than 70us or 700us.
- or, E2E latency 800us, and E2E jitter less than 80us or 800us.
- or, E2E latency 900us, and E2E jitter less than 90us or 900us.
- or, E2E latency 1000us, and E2E jitter less than 100us or 1ms.

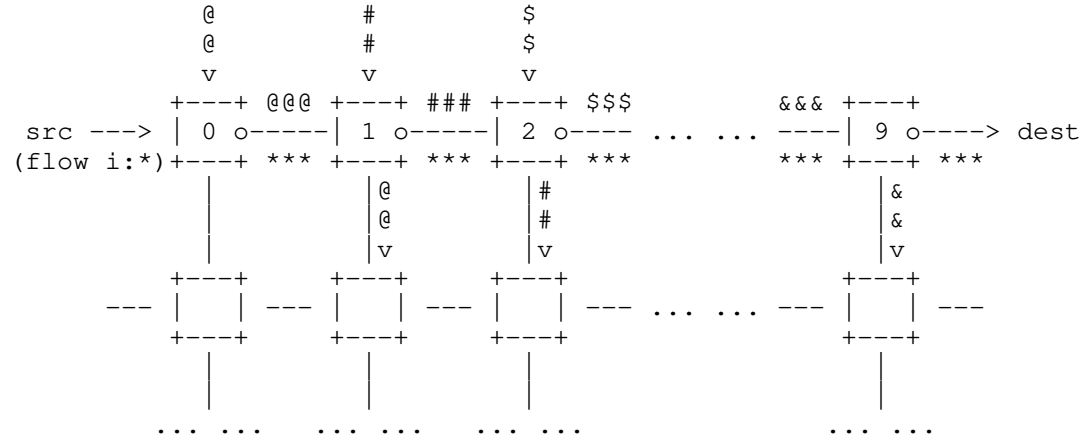

Figure 9: Heavyweight Loading Topology Example

For the observed flow i (marked with *), its TSpec and RSpec may be any of the above. Assuming that the path calculated by the controller for the flow i passes through 10 nodes (i.e., node 0~9). Especially, at each hop, flow i may conflict with other competitive flows, also with similar TSpec and RSpec as above, originated from other sources, e.g, competing with flow-set "@" at node 0, competing with flow-set "#" at node 1, etc.

For each link along the path, it may configure OPL-10ms instance with dedicated bandwidth 10 Gbps, containing 1000 timeslots each with length 10us. Assuming no link propagation delay and intra node forwarding delay, if flow i consumes outgoing timeslot by o=1, it can ensure an E2E latency of 100us (i.e., o * TL * 10 hops), and jitter of 20us(on-time mode) or 100us (in-time mode). The consumption by other o values is similar.

The table below shows the possible supported service scales. As flows arrived synchronously, the consumption of each timeslot in the orchestration period may be caused by any value of o. For example, if the ideal incoming timeslots of all flows are perfectly interleaved, then they can all consume timeslots by o=1 to get per-

hop latency 10us, or all consume timeslots by o=2 to get per-hop latency 20us, etc. However, due to the fixed length of OPL, after all timeslot resources are exhausted by specific o value, it means that there are no timeslot resources used by other o values. Another example is that the ideal incoming timeslots of all flows are the same, then some of them consume timeslots by o=1, some consume timeslots by o=2, and so on. In either case, the total service scale is OPL * C / burst_size, that is composed of sum(s_i), where s_i is the service scale for o = i. The table provides the total scale and the average scale corresponding to each o value.

Note that in the table each column only shows the data where all flows served based on all o values have the same TSpec (e.g, in the first colunm, TSpec per flow is burst size 1000 bits and arrival rate 1 Mbps), while in reality, flows served based on different o values generally have different TSpec. It is easy to add colunms to describe various combinations.

	o=1	o=2	o=3	o=4	o=5	o=6	o=7	o=8	o=9	o=10
TSpec: 1000 bits SBI 1 ms	total = 10000									
	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
TSpec: 1000 bits SBI 100 us	total = 1000									
	100	100	100	100	100	100	100	100	100	100
TSpec: 1000 bits SBI 10 us	total = 100									
	10	10	10	10	10	10	10	10	10	10
TSpec: 10000 bits SBI 10 ms	total = 10000									
	1000	1000	1000	1000	1000	1000	1000	1000	1000	1000
TSpec: 10000 bits SBI 1 ms	total = 1000									
	100	100	100	100	100	100	100	100	100	100
TSpec: 10000 bits SBI 100 us	total = 100									
	10	10	10	10	10	10	10	10	10	10

Figure 10: Timeslot Reservation and Service Scale Example

15.3.2. Lightweight Loading Examples

The following examples observe how the preset service scale is supported and mapped to different timeslots by the TQF mechanism in the lightweight loading case.

In these examples, the network only contains a small number of bottleneck links with low network utilization, and it can be considered as the lightweight loading case of Figure 9. Lightweight loading usually means having a smaller calculated worst-case latency per hop, or the actual latency experienced doesn't reach the worst-case latency.

15.3.2.1. Grid Reference Topology

[I-D.ietf-detnet-dataplane-taxonomy] describes a grid topology (Figure 11) with partial mesh. Three flow types, i.e., audio, video, and CC (Command and Control) are considered to require deterministic networking services. Among them, audio and CC flows consume less bandwidth (1.6 Mbps per flow and 0.48 Mbps per flow respectively) but both require lower E2E latency (5ms), while video flows consume more bandwidth (11 Mbps per flow) but can tolerate larger E2E latency (10ms).

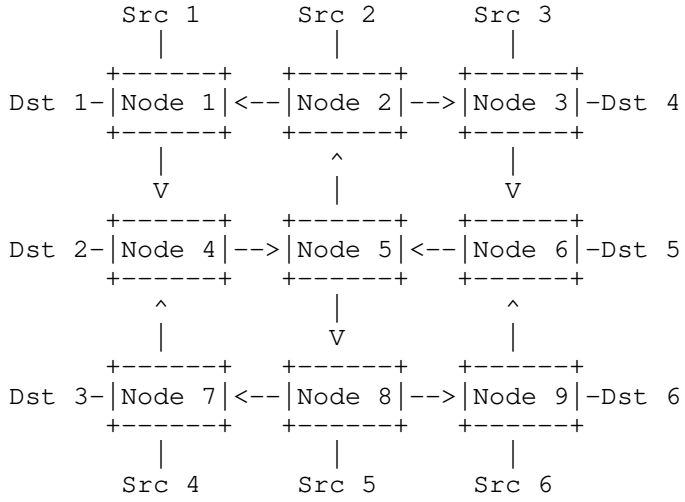

Figure 11: Lightweight Loading Topology Example

According to the preset rules that generate a unique route for every source and destination pair, the details of all paths are as follows:

```
Src1-1-Dst1
Src1-1-4-Dst2
Src1-1-4-5-8-7-Dst3
Src1-1-4-5-2-3-Dst4
Src1-1-4-5-8-9-6-Dst5
Src1-1-4-5-8-9-Dst6

Src2-2-1-Dst1
Src2-2-1-4-Dst2
Src2-2-3-6-5-8-7-Dst3
Src2-2-3-Dst4
Src2-2-3-6-Dst5
Src2-2-3-6-5-8-9-Dst6

Src3-3-6-5-2-1-Dst1
Src3-3-6-5-2-1-4-Dst2
Src3-3-6-5-8-7-Dst3
Src3-3-Dst4
Src3-3-6-Dst5
Src3-3-6-5-8-9-Dst6

Src4-7-4-5-2-1-Dst1
Src4-7-4-Dst2
Src4-7-Dst3
Src4-7-4-5-2-3-Dst4
Src4-7-4-5-8-9-6-Dst5
Src4-7-4-5-8-9-Dst6

Src5-8-7-4-5-2-1-Dst1
Src5-8-7-4-Dst2
Src5-8-7-Dst3
Src5-8-7-4-5-2-3-Dst4
Src5-8-9-6-Dst5
Src5-8-9-Dst6

Src6-9-6-5-2-1-Dst1
Src6-9-6-5-2-1-4-Dst2
Src6-9-6-5-8-7-Dst3
Src6-9-6-5-2-3-Dst4
Src6-9-6-Dst5
Src6-9-Dst6
```

Where, flows to destination Dst1 and Dst6 are audio flows, flows to destination Dst2 and Dst5 are CC flows, and flows to destination Dst3 and Dst4 are video flows. Each path carries 10 flows, e.g., the path "Src1-1-Dst1" carries 10 audio flows. It can be seen that the longest path contains 7 hops, and the bottleneck link involves link (2-3) and link (8-7), both of which have 10 audio flows, 60 video flows, and 10 CC flows.

Grid topology in this example only contains a small number of bottleneck links with low network utilization, and it can be considered as the lightweight loading case of Figure 9. Lightweight loading usually means having a smaller calculated worst-case latency per hop, or the actual latency experienced cannot reach the worst-case latency.

According to the longest path and the expected E2E latency, the per-hop latency bound for each type of flow can be estimated, i.e., 700us for audio and CC flows, 1400us for video flows. This means that the TQF mechanism needs to provide appropriate timeslots, and the offset between the incoming timeslots and outgoing timeslots mapped by flow audio, CC, and video, cannot be larger than 700us, 700us, 1400us, respectively.

A TQF instance with OPL-5ms may be configured on each link in the network. The reason for choosing OPL-5ms is that it is approximately the Least Common Multiple of the packet intervals of the three type of flows. In this example, the regulated packet interval (i.e., packet_size / service_rate) for flows audio, video, and CC are 1.25 ms, 1.1 ms, and 5 ms, respectively. So, within each OP, it contains 4 audio packets per flow, 5 video packets per flow, and 1 CC packet per flow. Taking an audio flow as an example, its four discrete packets will occupy four ideal positions in OP, and we denote these packets as four sub-bursts, namely sub-burst 1, 2, 3, and 4. Similarly, a CC flow will occupy one ideal position in OP, by sub-burst 1. And, a video flow will occupy five ideal position in OP, by sub-burst 1, 2, 3, 4, 5, respectively.

Considering the maximum packet size is 12000 bits (from video flow) and the link capacity is 1 Gbps, timeslot length 100us is selected to accommodate at least such a single packet. Intuitively, if the link capability is larger, such as 10 Gbps, the timeslot length can be chosen to be smaller, such as 10us.

Although the number of flows on different links varies, to simplify the description, the collection of various types of flows on bottleneck and non bottleneck links is taken as the number of flows on each link to calculate the timeslot resource requirements. So we slightly increase the loading and assume that the number of each type of flows on each link reaches 60.

Figure 12 shows a possible timeslots mapped by flows on a link. Note that it assumes that the first sub-burst of all flows (i.e., 60 CC flows, 60 audio flows, and 60 video flows) arrive concurrently at the same time T0 (nearly before timeslot #0) and some packets have to experience larger per-hop latency than others, and this is the worst case. In fact, all flows may arrive in different timeslots interleaved, so that the consumed outgoing timeslots are also naturally interleaved and each packet may experience smaller per-hop latency (e.g., one timeslot length).

Slot	Bursts	Services Mapped
#0	100 Kbits	40 CC flows(sub-burst 1), 96 Kbits 2 audio flows(sub-burst 1), 4 Kbits
#1	100 Kbits	20 CC flows(sub-burst 1), 48 Kbits 26 audio flows(sub-burst 1), 52 Kbits
#2	100 Kbits	26 audio flows(sub-burst 1), 52 Kbits 4 video flows(sub-burst 1), 48 Kbits
#3	100 Kbits	8 video flows(sub-burst 1), 96 Kbits 2 audio flows(sub-burst 1), 4 Kbits
#4	100 Kbits	8 video flows(sub-burst 1), 96 Kbits 2 audio flows(sub-burst 1), 4 Kbits
#5	100 Kbits	8 video flows(sub-burst 1), 96 Kbits 2 audio flows(sub-burst 1), 4 Kbits
#6	100 Kbits	8 video flows(sub-burst 1), 96 Kbits remaining 4 Kbits
#7	100 Kbits	8 video flows(sub-burst 1), 96 Kbits remaining 4 Kbits
#8	100 Kbits	8 video flows(sub-burst 1), 96 Kbits remaining 4 Kbits

#9	100 Kbits	8 video flows(sub-burst 1), 96 Kbits remaining 4 Kbits
#10	100 Kbits	
#11	100 Kbits	8 video flows(sub-burst 2), 96 Kbits remaining 4 Kbits
#12	100 Kbits	8 video flows(sub-burst 2), 96 Kbits remaining 4 Kbits
#13	100 Kbits	50 audio flows(sub-burst 2), 100K bits
#14	100 Kbits	10 audio flows(sub-burst 2), 20 Kbits 4 video flows(sub-burst 2), 48 Kbits remaining 32 Kbits
#15	100 Kbits	8 video flows(sub-burst 2), 96 Kbits remaining 4 Kbits
#16	100 Kbits	8 video flows(sub-burst 2), 96 Kbits remaining 4 Kbits
#17	100 Kbits	8 video flows(sub-burst 2), 96 Kbits remaining 4 Kbits
#18	100 Kbits	8 video flows(sub-burst 2), 96 Kbits remaining 4 Kbits
#19	100 Kbits	8 video flows(sub-burst 2), 96 Kbits remaining 4 Kbits
#20	100 Kbits	
#21	100 Kbits	8 video flows(sub-burst 3), 96 Kbits remaining 4 Kbits
#22	100 Kbits	8 video flows(sub-burst 3), 96 Kbits remaining 4 Kbits
#23	100 Kbits	8 video flows(sub-burst 3), 96 Kbits remaining 4 Kbits
#24	100 Kbits	8 video flows(sub-burst 3), 96 Kbits remaining 4 Kbits
#25	100 Kbits	8 video flows(sub-burst 3), 96 Kbits remaining 4 Kbits

#26	100 Kbits	50 audio flows(sub-burst 2), 100 Kbits
#27	100 Kbits	10 audio flows(sub-burst 3), 20 Kbits 4 video flows(sub-burst 3), 48 Kbits remaining 32 Kbits
#28	100 Kbits	8 video flows(sub-burst 3), 96 Kbits remaining 4 Kbits
#29	100 Kbits	8 video flows(sub-burst 3), 96 Kbits remaining 4 Kbits
#30	100 Kbits	
#31	100 Kbits	4 video flows(sub-burst 4), 48 Kbits remaining 52Kbits
#32	100 Kbits	8 video flows(sub-burst 4), 96 Kbits remaining 4 Kbits
#33	100 Kbits	8 video flows(sub-burst 4), 96 Kbits remaining 4 Kbits
#34	100 Kbits	8 video flows(sub-burst 4), 96 Kbits remaining 4 Kbits
#35	100 Kbits	8 video flows(sub-burst 4), 96 Kbits remaining 4 Kbits
#36	100 Kbits	8 video flows(sub-burst 4), 96 Kbits remaining 4 Kbits
#37	100 Kbits	8 video flows(sub-burst 4), 96 Kbits remaining 4 Kbits
#38	100 Kbits	8 video flows(sub-burst 4), 96 Kbits remaining 4 Kbits
#39	100 Kbits	50 audio flows(sub-burst 4), 100 Kbits
#40	100 Kbits	10 audio flows(sub-burst 4), 20 Kbits remaining 80 Kbits
#41	100 Kbits	
#42	100 Kbits	4 video flows(sub-burst 5), 48 Kbits remaining 52Kbits

#43	100 Kbits	8 video flows(sub-burst 5), 96 Kbits remaining 4 Kbits
#44	100 Kbits	8 video flows(sub-burst 5), 96 Kbits remaining 4 Kbits
#45	100 Kbits	8 video flows(sub-burst 5), 96 Kbits remaining 4 Kbits
#46	100 Kbits	8 video flows(sub-burst 5), 96 Kbits remaining 4 Kbits
#47	100 Kbits	8 video flows(sub-burst 5), 96 Kbits remaining 4 Kbits
#48	100 Kbits	8 video flows(sub-burst 5), 96 Kbits remaining 4 Kbits
#49	100 Kbits	8 video flows(sub-burst 5), 96 Kbits remaining 4 Kbits

Figure 12: Timeslot Resource Pool and Service Mapped in Grid Topology

Each CC flow contributes only one sub-burst within the OP:

* The first sub-bursts of all 60 CC flows, totaling 144 Kbits, arrived nearly before timeslot #0, consume timeslots #0, #1. The worst-case per-hop latecy for the last packet sent is 200 us.

Each audio flow contributes four sub-burst within the OP:

* The first sub-bursts of all audio flows, totaling 120 Kbits, arrived nearly before timeslot #0, consume timeslots #0, #1, #2, #3, #4, #5. The worst-case per-hop latecy for the last packet sent is 600 us.

* The second sub-bursts of all audio flows, totaling 120 Kbits, arrived nearly before timeslot #13, consume timeslot #13, #14. The worst-case per-hop latecy for the last packet sent is 200 us.

* The third sub-bursts of all audio flows, totaling 120 Kbits, arrived nearly before timeslot #26, consume timeslots #26, #27. The worst-case per-hop latecy for the last packet sent is 200 us.

* The fourth sub-bursts of all audio flows, totaling 120 Kbits, arrived nearly before timeslot #39, consume timeslots #39, #40. The worst-case per-hop latecy for the last packet sent is 200 us.

Each video flow contributes five sub-burst within the OP:

* The first sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #0, consume timeslots #2, #3, #4, #5, #6, #7, #8, #9. The worst-case per-hop latecy for the last packet sent is 1000 us.

* The second sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #11, consume timeslots #11, #12, #14, #15, #16, #17, #18, #19. The worst-case per-hop latecy for the last packet sent is 900 us.

* The third sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #21, consume timeslots #21, #22, #23, #24, #25, #27, #28, #29. The worst-case per-hop latecy for the last packet sent is 900 us.

* The fourth sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #31, consume timeslots #31, #32, #33, #34, #35, #36, #37, #38. The worst-case per-hop latecy for the last packet sent is 900 us.

* The fifth sub-bursts of all video flows, totaling 720 Kbits, arrived nearly before timeslot #41, consume timeslots #42, #43, #44, #45, #46, #47, #48, #49. The worst-case per-hop latecy for the last packet sent is 900 us.

NOTE:

* In the above process of resource allocation, the 10 flows carried on each path are individually allocated burst resources. This is the most general case, that is, although the 10 flows share the same path, they are assumed to be independent of each other. However, in some cases, if these 10 flows are treated as a macro flow and policing is executed at the network entrance node for the macro flow (the leaky bucket depth is still the maximum packet size, but the leack bucket rate is the aggregation rate), and resources are reserved for the macro flow instead of the member flow, then it is still necessary to allocate timeslot resources for the same total amounts of bursts, which will be more evenly distributed (i.e., more interleaved) within the OP. For example, a macro CC flow (including 10 CC member flows) contribute 10 sub-bursts within the OP (5 ms), and each sub-burst size is 2400 bits.

* Video flows have 30 back-to-back packets per single burst, and are being regulated on the flow entrance node, to support 60 video flows on each link. Operators may increase the bucket depth for video flows to make the shaped pattern and the original arrival pattern as consistent as possible, but this will be harmful to service scale. There is a tradeoff between burstiness, policing, and service scale.

15.3.2.2. Ring-Mesh Reference Topology

[I-D.ietf-detnet-dataplane-taxonomy] describes another hierarchical Ring-Mesh topology (Figure 13), where, node 1~9 are core routers, and each leaf group consists of 10 Ring networks. Each Ring network (Figure 14) has 8 nodes, with one node connected to the core by a separate inter-domain link.

The capacity of all the links in the core network is 10 Gbps. The capacity of all the links in the leaf network, including the inter-domain link, is 1 Gbps.

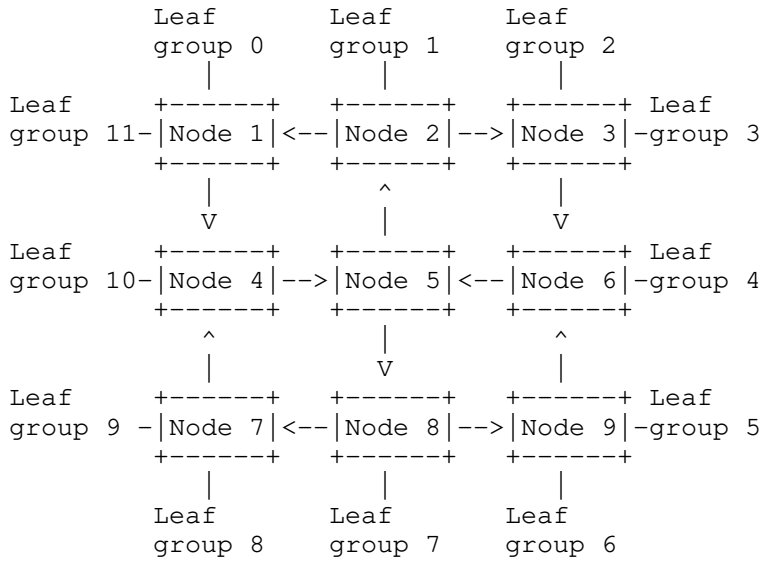

Figure 13: Hierarchical Ring-Mesh Topology

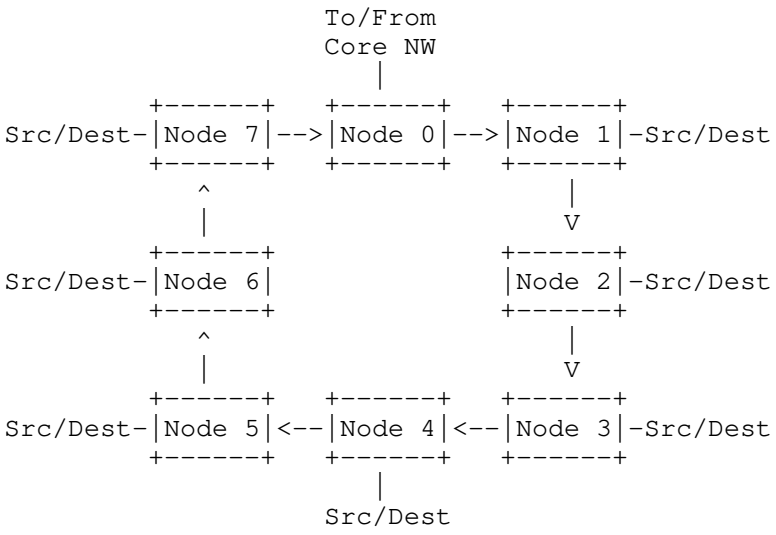

Figure 14: Ring Network

Again, three flow types, i.e., audio, video, and CC (Command and Control) are considered to require deterministic networking services, whose TSpec and RSpec are consistent with the above Grid example.

A flow-set is defined that includes 7 audio flows, 7 video flows, and 32 CC flows, all of which share the same DetNet path. For example, node 1 in the source ring may send a flow-set to node 7 in the destination ring, and the DetNet path may be, 1-2-3-4-5-6-7-0 (source ring), inter-domain link, core, inter-domain link, 0-1-2-3-4-5-6-7 (destination ring).

The longest DetNet path may be 20 hops, where, 7 hops in the source ring, 7 hops in the destination ring, 2 inter-domain hops, and 4 hops in the core.

The preset routing of DetNet path is that, every flow-set in a ring network travels from node i to node (i+7)mod8, and each leaf group i sends n flow-sets (e.g., n = 10, if a leaf group contains 10 ring networks) to the leaf group (i+6)mod12.

Take a flow-set from the source ring to the destination ring as the observed flow-set. The observed flow-set will compete with other 6 flow-sets in the ring, and compete with more flow-sets (coming from other leaf groups) in the core. Note that there is no competition on the inter-domain link.

In this example, we no longer assume that every packet of all flow-sets, including the observed flow-set and the competed flow-sets, arrives simultaneously. Although assuming extremely high concurrency can accommodate any topology with some actual concurrency, it underestimate the service scale that can be admitted. In fact, in the ring network, the concurrency at each hop is that only two input interfaces compete for one output interface. For inter-domain links, concurrency is even zero. In the core network, concurrency is also limited. By utilizing the knowledge of concurrency, more reasonable allocation of slots can be applied for all flows.

A TQF instance OPL-5ms with timeslot length 100us is configured on each link in both ring and core networks.

In the ring network, a bad flow interleaving is that each node generates a flow-set at the same time (e.g., nearly before timeslot #0). Assuming that each node prioritizes obtaining a smaller timeslot offset for locally generated flow-set, i.e., each node allocates timeslots for the local flow-set starting from #0, while allocates later timeslots for the flow-sets received from upstream nodes. When a local flow-set, e.g., f1 generated by Node 1, is sent to the downstrem Node 2, it will conflict with another local flow-set, e.g., f2 generated by Node 2, and has to be assigned a later timeslot (i.e., obtaining a larger timeslot offset). Since that f1 is assigned a later outgoing timeslot on Node 2 means that the incoming timeslot on Node 3 is also later, it does not cause f1 to face more queuing delay on node 3 (and also other downstream nodes). Therefore, as the number of hops increases, a perfect flow interleaving is formed. For example, on Node 1, the interleaved flow-sets received may be {f0, f7, f6, f5, f4, f3, f2}, among which f0 (generated by Node 0) arrived the earliest and f2 (generated by Node 2) arrived the latest. Note that on Node 1, f2 will be terminated and not compete for outgoing port.

Figure 15 shows a possible timeslots mapped by flows on link-1-2 in the ring domain. Other links are similar to link-1-2. For a flow-set, try to allocate smaller timeslot offsets for CC, then audio, and at last video, as video flows have loose latency requirements. For simplicity, assuming no link propagation delay and intra node forwarding delay.

Slot	Bursts	Services Mapped
#0	100 Kbits	32 CC flows of f1 9 CC flows of f0

#1	100 Kbits	23 CC flows of f0 18 CC flows of f7
#2	100 Kbits	14 CC flows of f7 27 CC flows of f6
#3	100 Kbits	5 CC flows of f6 32 CC flows of f5 4 CC flows of f4
#4	100 Kbits	28 CC flows of f4 13 CC flows of f3
#5	100 Kbits	19 CC flows of f3 each 7 audio flows of f1, f0, f7 (round 1) 6 audio flows of f6 (round 1)
#6	100 Kbits	1 audio flows of f6 (round 1) each 7 audio flows of f5, f4, f3 (round 1) 4 video flows of f1 (round 1)
#7	100 Kbits	3 video flows of f1 (round 1) 5 video flows of f0 (round 1)
#8	100 Kbits	2 video flows of f0 (round 1) 6 video flows of f7 (round 1)
#9	100 Kbits	1 video flows of f7 (round 1) 7 video flows of f6 (round 1)
#10	100 Kbits	7 video flows of f5 (round 1) 1 video flows of f4 (round 1)
#11	100 Kbits	6 video flows of f4 (round 1) 2 video flows of f3 (round 1)
#12	100 Kbits	5 video flows of f3 (round 1) 3 video flows of f1 (round 2)
#13	100 Kbits	each 7 audio flows of f1, f0, f7~f3 (round 2)
#14	100 Kbits	4 video flows of f1 (round 2) 4 video flows of f0 (round 2)
#15	100 Kbits	3 video flows of f0 (round 2) 5 video flows of f7 (round 2)
#16	100 Kbits	2 video flows of f7 (round 2)

		6 video flows of f6 (round 2)
#17	100 Kbits	1 video flows of f6 (round 2) 7 video flows of f5 (round 2)
#18	100 Kbits	7 video flows of f4 (round 2) 1 video flows of f3 (round 2)
#19	100 Kbits	6 video flows of f3 (round 2)
#20	100 Kbits	
#21	100 Kbits	7 video flows of f1 (round 3) 1 video flows of f0 (round 3)
#22	100 Kbits	6 video flows of f0 (round 3) 2 video flows of f7 (round 3)
#23	100 Kbits	5 video flows of f7 (round 3) 3 video flows of f6 (round 3)
#24	100 Kbits	4 video flows of f6 (round 3) 4 video flows of f5 (round 3)
#25	100 Kbits	3 video flows of f5 (round 3) 5 video flows of f4 (round 3)
#26	100 Kbits	each 7 audio flows of f1, f0, f7~f3 (round 3)
#27	100 Kbits	2 video flows of f4 (round 3) 6 video flows of f3 (round 3)
#28	100 Kbits	1 video flows of f3 (round 3)
#29	100 Kbits	
#30	100 Kbits	
#31	100 Kbits	7 video flows of f1 (round 4) 1 video flows of f0 (round 4)
#32	100 Kbits	6 video flows of f0 (round 4) 2 video flows of f7 (round 4)
#33	100 Kbits	5 video flows of f7 (round 4) 3 video flows of f6 (round 4)
#34	100 Kbits	4 video flows of f6 (round 4)

		4 video flows of f5 (round 4)
#35	100 Kbits	3 video flows of f5 (round 4) 5 video flows of f4 (round 4)
#36	100 Kbits	2 video flows of f4 (round 4) 6 video flows of f3 (round 4)
#37	100 Kbits	1 video flows of f3 (round 4)
#38	100 Kbits	
#39	100 Kbits	each 7 audio flows of f1, f0, f7~f3 (round 4)
#40	100 Kbits	
#41	100 Kbits	7 video flows of f1 (round 5) 1 video flows of f0 (round 5)
#42	100 Kbits	6 video flows of f0 (round 5) 2 video flows of f7 (round 5)
#43	100 Kbits	5 video flows of f7 (round 5) 3 video flows of f6 (round 5)
#44	100 Kbits	4 video flows of f6 (round 5) 4 video flows of f5 (round 5)
#45	100 Kbits	3 video flows of f5 (round 5) 5 video flows of f4 (round 5)
#46	100 Kbits	2 video flows of f4 (round 5) 6 video flows of f3 (round 5)
#47	100 Kbits	1 video flows of f3 (round 5)
#48	100 Kbits	
#49	100 Kbits	

Figure 15: Timeslot Resource Pool and Service Mapped in Ring Domain

Each CC flow contributes only one round within the OP:

* The first round of all 7*32 CC flows, totaling 537.6 Kbits, arrive sequentially in the duration from timeslot #0 to #5, consuming timeslots #0 to #5 respectively. The worst-case per-hop latecy is 100 us.

Each audio flow contributes four sub-burst within the OP:

* The first round of all 7*7 audio flows, totaling 98 Kbits, arrive sequentially in the duration from timeslot #5 to #6, except that the local generated 7 audio flows arrive at timeslot #0, consuming timeslots #5 to #6 respectively. The worst-case per-hop latecy is 500 us for the local generated 7 audio flows, and 100 us for other audio flows.

* The second round of all 7*7 audio flows, totaling 98 Kbits, arrive sequentially in the duration of timeslot #13, consuming timeslots #13 respectively. The worst-case per-hop latecy is 100 us.

* The third round of all 7*7 audio flows, totaling 98 Kbits, arrive sequentially in the duration of timeslot #26, consuming timeslots #26 respectively. The worst-case per-hop latecy is 100 us.

* The fourth round of all 7*7 audio flows, totaling 98 Kbits, arrive sequentially in the duration of timeslot #39, consuming timeslots #39 respectively. The worst-case per-hop latecy is 100 us.

Each video flow contributes five sub-burst within the OP:

* The first round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #6 to #12, except that the local generated 7 video flows arrive at timeslot #0, consuming timeslots #6 to #12 respectively. The worst-case per-hop latecy is 600 us for the local generated 7 video flows, and 200 us for other audio flows.

* The second round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #12 to #19, except that the local generated 7 video flows arrive at timeslot #11, consuming timeslots #12 to #19 respectively. The worst-case per-hop latecy is 300 us for the local generated 7 video flows, and 200 us for other audio flows.

* The third round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #21 to #27, consuming timeslots #21 to #27 respectively. The worst-case per-hop latecy is 200 us.

* The fourth round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #31 to #37, consuming timeslots #31 to #37 respectively. The worst-case per-hop latecy is 200 us.

* The fifth round of all 7*7 video flows, totaling 588 Kbits, arrive sequentially in the duration from timeslot #41 to #47, consuming timeslots #41 to #47 respectively. The worst-case per-hop latecy is 200 us.

In the core network, the details of all DetNet paths are as follows:

```
group0 -1-4-5-8-9- group6
group1 -2-1-4-5-8- group7
group2 -3-6-5-8-7- group8
group3 -3-6-5-8-7- group9
group4 -6-5-2-1-4- group10
group5 -9-6-5-2-1- group11
group6 -9-6-5-2-1- group0
group7 -8-7-4-5-2- group1
group8 -7-4-5-2-3- group2
group9 -7-4-5-2-3- group3
group10 -4-5-2-3-6- group4
group11 -1-4-5-8-9- group5
```

Where, the bottleneck link-4-5 will carry 70 flow-sets, in which, 10 flow-sets each from separate inter-domain link, 30 flow-sets from node 1, and 30 flow-sets from node 7.

Another bottleneck link-5-2 will also carry 70 flow-sets,in which, 30 flow-set from node 6, and 40 flow-sets from node 4.

On the bottleneck link-4-5, a bad flow interleaving is that there are 12 bursts competing for the outgoing interface. Their sizes are 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 30, and 30 flow-sets respectively. Assuming that each incoming flow-set arrives nearly before timeslot #0. A simple timeslot resource allocation method similar to Section 15.3.2.1 can be used (assuming extreme concurrency), see Figure 16. In fact, all flows may arrive in different timeslots interleaved, so that the consumed outgoing timeslots are also naturally interleaved and each packet may experience smaller per-hop latency (e.g., one timeslot length), that is similar to the method in ring domain.

Slot	Bursts	Services Mapped
#0	1 Mbits	416 CC flows
#1	1 Mbits	416 CC flows
#2	1 Mbits	416 CC flows
#3	1 Mbits	416 CC flows
#4	1 Mbits	416 CC flows
#5	1 Mbits	160 CC flows 308 audio flows (round 1)
#6	1 Mbits	182 audio flows (round 1) 53 video flows (round 1)
#7	1 Mbits	83 video flows (round 1)
#8	1 Mbits	83 video flows (round 1)
#9	1 Mbits	83 video flows (round 1)
#10	1 Mbits	83 video flows (round 1)
#11	1 Mbits	83 video flows (round 1)
#12	1 Mbits	22 video flows (round 1) 61 video flows (round 2)
#13	1 Mbits	490 audio flows (round 2)
#14	1 Mbits	83 video flows (round 2)
#15	1 Mbits	83 video flows (round 2)
#16	1 Mbits	83 video flows (round 2)
#17	1 Mbits	83 video flows (round 2)
#18	1 Mbits	83 video flows (round 2)
#19	1 Mbits	14 video flows (round 2)
#20	1 Mbits	

#21	1 Mbits	83 video flows (round 3)
#22	1 Mbits	83 video flows (round 3)
#23	1 Mbits	83 video flows (round 3)
#24	1 Mbits	83 video flows (round 3)
#25	1 Mbits	83 video flows (round 3)
#26	1 Mbits	75 video flows (round 3) 50 audio flows (round 3)
#27	1 Mbits	440 audio flows (round 3)
#28	1 Mbits	
#29	1 Mbits	
#30	1 Mbits	
#31	1 Mbits	83 video flows (round 4)
#32	1 Mbits	83 video flows (round 4)
#33	1 Mbits	83 video flows (round 4)
#34	1 Mbits	83 video flows (round 4)
#35	1 Mbits	83 video flows (round 4)
#36	1 Mbits	75 video flows (round 4)
#37	1 Mbits	
#38	1 Mbits	
#39	1 Mbits	490 audio flows (round 4)
#40	1 Mbits	
#41	1 Mbits	83 video flows (round 5)
#42	1 Mbits	83 video flows (round 5)
#43	1 Mbits	83 video flows (round 5)
#44	1 Mbits	83 video flows (round 5)

#45	1 Mbits	83 video flows (round 5)
#46	1 Mbits	75 video flows (round 5)
#47	1 Mbits	
#48	1 Mbits	
#49	1 Mbits	

Figure 16: Timeslot Resource Pool and Service Mapped in Core Domain

According to the above table, the worst-case per-hop latecy is 600 us for CC, 700 us for audio, 1.3 ms for Video.

Other explanations are similar to the previous example of Grid reference topology.

16. IANA Considerations

TBD.

17. Security Considerations

Security considerations for DetNet are described in detail in [RFC9055]. General security considerations for the DetNet architecture are described in [RFC8655]. Considerations specific to the DetNet data plane are summarized in [RFC8938].

Adequate admission control policies should be configured in the edge of the DetNet domain to control access to specific timeslot resources. Access to classification and mapping tables must be controlled to prevent misbehaviors, e.g, an unauthorized entity may modify the table to map traffic to an unallowed timeslot resource, and competes and interferes with normal traffic.

18. Acknowledgements

TBD.

19. References

19.1. Normative References

[I-D.chen-detnet-sr-based-bounded-latency]
Chen, M., Geng, X., Li, Z., Joung, J., and J. Ryoo, "Segment Routing (SR) Based Bounded Latency", Work in Progress, Internet-Draft, draft-chen-detnet-sr-based-bounded-latency-03, 7 July 2023, <https://datatracker.ietf.org/doc/html/draft-chen-detnet-sr-based-bounded-latency-03>.

[I-D.eckert-detnet-tcqf]
Eckert, T. T., Li, Y., Bryant, S., Malis, A. G., Ryoo, J., Liu, P., Li, G., Ren, S., and F. Yang, "Deterministic Networking (DetNet) Data Plane - Tagged Cyclic Queuing and Forwarding (TCQF) for bounded latency with low jitter in large scale DetNets", Work in Progress, Internet-Draft, draft-eckert-detnet-tcqf-07, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-eckert-detnet-tcqf-07>.

[I-D.ietf-detnet-dataplane-taxonomy]
Joung, J., Geng, X., Peng, S., and T. T. Eckert, "Dataplane Enhancement Taxonomy", Work in Progress, Internet-Draft, draft-ietf-detnet-dataplane-taxonomy-03, 2 March 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-dataplane-taxonomy-03>.

[I-D.ietf-detnet-scaling-requirements]
Liu, P., Li, Y., Eckert, T. T., Xiong, Q., Ryoo, J., zhushiyin, and X. Geng, "Requirements for Scaling Deterministic Networks", Work in Progress, Internet-Draft, draft-ietf-detnet-scaling-requirements-07, 20 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-scaling-requirements-07>.

[I-D.p-6man-deterministic-eh]
Peng, S., "Deterministic Source Route Header", Work in Progress, Internet-Draft, draft-p-6man-deterministic-eh-01, 10 October 2024, <https://datatracker.ietf.org/doc/html/draft-p-6man-deterministic-eh-01>.

[I-D.pb-6man-deterministic-crh]
Peng, S. and R. Bonica, "Deterministic Routing Header", Work in Progress, Internet-Draft, draft-pb-6man-deterministic-crh-01, 10 October 2024, <https://datatracker.ietf.org/doc/html/draft-pb-6man-deterministic-crh-01>.

[I-D.peng-lsr-deterministic-traffic-engineering]
Peng, S., "IGP Extensions for Deterministic Traffic Engineering", Work in Progress, Internet-Draft, draft-peng-lsr-deterministic-traffic-engineering-03, 23 December 2024, <https://datatracker.ietf.org/doc/html/draft-peng-lsr-deterministic-traffic-engineering-03>.

[I-D.xp-ippm-detnet-stamp]
Min, X., Peng, S., and X. hexiaoming, "STAMP Extensions for DetNet", Work in Progress, Internet-Draft, draft-xp-ippm-detnet-stamp-01, 16 December 2024, <https://datatracker.ietf.org/doc/html/draft-xp-ippm-detnet-stamp-01>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC8938] Varga, B., Ed., Farkas, J., Berger, L., Malis, A., and S. Bryant, "Deterministic Networking (DetNet) Data Plane Framework", RFC 8938, DOI 10.17487/RFC8938, November 2020, <https://www.rfc-editor.org/info/rfc8938>.

[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>.

19.2. Informative References

[ATM-LATENCY]
"Bounded Latency Scheduling Scheme for ATM Cells", 1999, <https://ieeexplore.ieee.org/document/780828/>.

[CQF] "Cyclic queueing and Forwarding", 2017, <https://ieeexplore.ieee.org/document/7961303>.

[ECQF] "Enhancements to Cyclic Queuing and Forwarding", 2023, <https://1.ieee802.org/tsn/802-1qdv/>.

[IEEE-1588] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems", 2008, <https://standards.ieee.org/findstds/standard/1588-2008.html>.

[SP-LATENCY] "Guaranteed Latency with SP", 2020, <https://ieeexplore.ieee.org/document/9249224>.

[syncE] "Timing and synchronization aspects in packet networks", 2013, <https://www.itu.int/rec/T-REC-G.8261>.

[TAS] "Time-Aware Shaper", 2015, <https://standards.ieee.org/ieee/802.1Qbv/6068/>.

Authors' Addresses

Shaofu Peng
ZTE
China
Email: peng.shaofu@zte.com.cn

Peng Liu
China Mobile
China
Email: liupengyjy@chinamobile.com

Kashinath Basu
Oxford Brookes University
United Kingdom
Email: kbasu@brookes.ac.uk

Aihua Liu
ZTE
China
Email: liu.aihua@zte.com.cn

Dong Yang
Beijing Jiaotong University
China

Email: dyang@bjtu.edu.cn

Guoyu Peng
Beijing University of Posts and Telecommunications
China
Email: guoyupeng@bupt.edu.cn

DetNet Working Group
Internet-Draft
Intended status: Standards Track
Expires: 1 September 2025

Y. Ryoo
ETRI
J. Joung
Sangmyung University

28 February 2025

Non-work Conserving Stateless Core Fair Queuing
draft-ryoo-detnet-nscore-00

Abstract

This document specifies the framework and operational procedure for deterministic networking that guarantees maximum and minimum end-to-end latency bounds to flows. The solution has non-periodic, asynchronous, flow-level, non-work conserving, on-time, and rate-based functional characteristics, according to the taxonomy suggested by [draft-ietf-detnet-dataplane-taxonomy-02].

The packets are stored in the queue in ascending order of the ideal service start time, called Eligible Time (ET), and the ideal service completion time, called Finish Time (FT). The queued packets were forwarded between ET and FT in a non-work conserving manner. The ET and FT are calculated at the entrance node according to the packet size and rate of the flow. All subsequent core nodes are stateless and asynchronously compute ET and FT based on metadata received via packet headers. This mechanism is called non-work-preserving stateless fair queuing, which guarantees both E2E latency upper and lower bounds.

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 1 September 2025.

Table of Contents

1. Introduction

A class of schedulers called Fair Queuing (FQ) limits interference between flows to the degree of the maximum packet size. In FQ, the ideal service completion time, called Finish Time (FT), of a packet is obtained from an imaginary system that can provide the ideal flow isolation. Applying this technique, the end-to-end (E2E) latency bound of a flow is similar to that of an ideally isolated system.

Since calculating the FT of the current packet requires the FT of previous packets within the flow, this means that nodes must manage the state of the flow. The complexity of managing the state of a large number of flows can be a burden, so the proposed framework for large-scale deterministic networking is called work conserving

stateless core fair queuing (C-SCORE), which generates FT for packets at the entrance node and marks FT in the packet to operate with stateless in core nodes.

However, C-SCORE is a scheduler of work conserving approach, so it has an in-time characteristic. Therefore, this draft proposes a non-work conserving scheduler method by extending C-SCORE to have an on-time characteristic, called N-SCORE. The entrance node additionally obtains an ideal service start time, called an eligible time (ET), of the current packet based on the FT of the previous packet or the arrival time of the current packet. All of the nodes queued packets in ascending order of the ET and FT and forward the packet between ET and FT in a non-work conserving approach. N-SCORE is a method that guarantees not only the upper bound but also the lower bound of E2E latency by adding ET while using the information managed by the entrance node of the existing C-SCORE.

2. Terminology

2.1. Symbols Used in This Document

FQ	fair queuing
FT	finish time
ET	eligible time
Fh(p)	FT of the packet p at the node h
Eh(p)	ET of the packet p at the node h
Ah(p)	arrinal time of the packet p at the node h
dh(p)	maximum delay of the packet p at the node h
ch(p)	service complition time of packet p at the node h
r(p)	service rate of the packet p
L(p)	length of the packet p
Rh	link capacity of the node h
Lhmax	maximum packet length of the node h
PDh	propagation delay of the link h

2.2. Abbreviations

3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. N-SCORE Packet Scheduler Framework

Utilizing the concept of virtual clock (VC) scheduler, C-SCORE defines FT for packet p as

$$F(p) = \max\{F(p-1), A(p)\} + L(p)/r(p). \quad (1)$$

Where (p-1) and p are consecutive packets of the flow being observed, F(p-1) is the finish time of p-1, A(p) is the arrival time of p, L(p) is the length of p, and r(p) is the flow service rate. Flow exponents are omitted.

In C-SCORE, the entrance node manages F(p-1) and obtains F(p) by comparing it with A(p). Then, it calculates F(p) of the next node and marks it in the packet header. The service period of packet p in each node is defined as (A(p), F(p)]. Assuming the link propagation delay is zero, an example of the packet service period at the entrance node and core node with the C-SCORE scheduler is illustrated as follows:

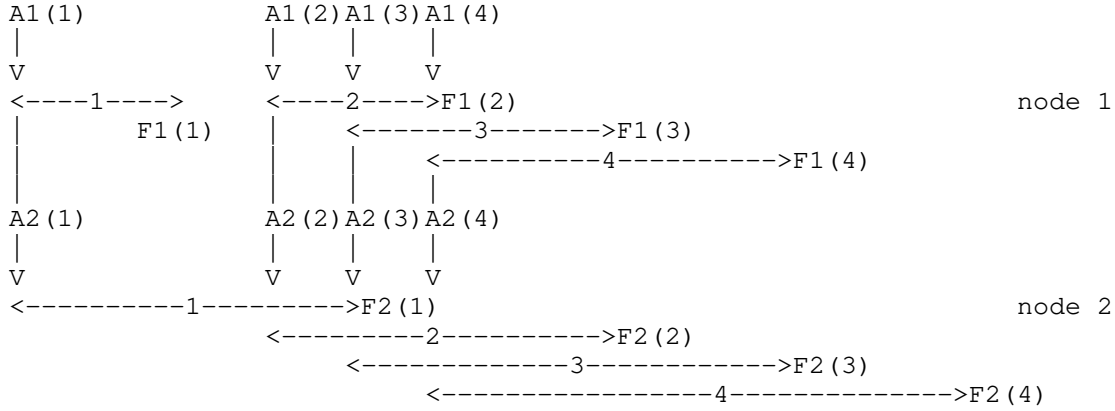

Figure 1: C-SCORE packet scheduler service period

The proposed N-SCORE framework introduces an additional parameter, ET (Eligible Time), which is used as the earliest possible packet service start time. Without requiring additional state management for ET, N-SCORE utilizes the information already managed by the entrance node in the existing C-SCORE to obtain ET and FT as follows:

$$E(p) = \max\{F(p-1), A(p)\} \quad (2)$$

$$F(p) = E(p) + L(p)/r \quad (3)$$

A packet can join the output link scheduler immediately after its ET. If no other packet is present in the scheduler, the packet is served right away. Otherwise, the packet joins the queue. Packets in the queue are served in ascending order of their ET and FT. Since the FT of N-SCORE is identical to that of C-SCORE, packets in N-SCORE follow the same service order as in C-SCORE. The only difference between the two systems is the existence of ET. However, in N-SCORE, due to the presence of the ET, the service period of packet p, while maintaining the same service order, is defined as (E(p), F(p)]. Here, E(p) and F(p) are ET and FT of packet p, respectively. Consequently, N-SCORE forwards packets in a non-work-preserving manner, maintaining a constant interval between E(p) and F(p) in all nodes. The service periods of packets within the same flow do not overlap at each node. Assuming zero link propagation delay, the packet service period at the entrance and core nodes with the N-SCORE scheduler is illustrated as follows:

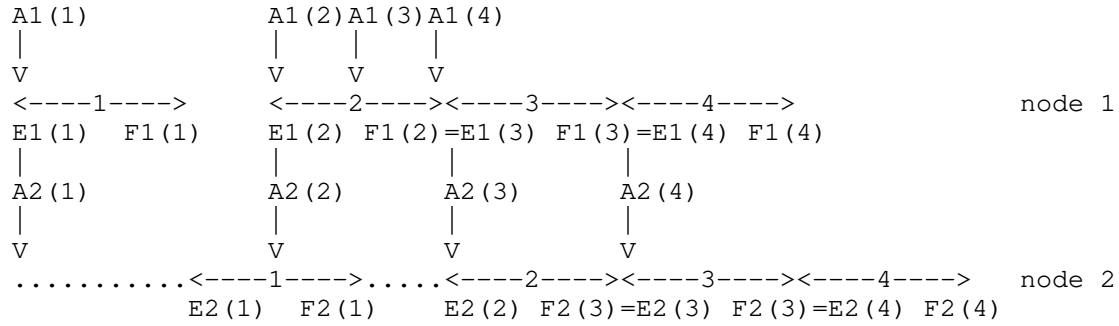

Figure 2: N-SCORE packet scheduler service period

5. E2E latency and jitter bound

The end-to-end (E2E) latency of N-SCORE is upper-bounded by:

(B-L)/r+â\210\221[h=0,H]{L/r + Lhmax/Rh} (4)

which is the same as that of C-SCORE, which operates based on FT. Here, B, L, and r represent the maximum burst size, maximum packet length, and service rate of the observed flow, respectively. The link propagation delay is omitted.

Unlike C-SCORE, which has no lower bound for E2E latency, the E2E latency of N-SCORE, which operates based on both ET and FT, is lower-bounded by:

â\210\221[h=0, H-1]{L/r+(Lhmax)/Rh} +Lmin/RH (5)

where L, Lmin, and r denote the maximum packet length, minimum packet length, and service rate of the observed flow, respectively. The link propagation delay is omitted.

Therefore, unlike C-SCORE, which exhibits high jitter ranging from 0 to the E2E maximum delay, the E2E jitter of N-SCORE is bounded by:

B/r+(LHmax)/RH - Lmin/RH (6)

6. Operational Procedure

The N-SCORE scheduler in all nodes has a deterministic service period of (E(p), F(p)] for packet p. Packets are queued in a priority queue in ascending order of ET and FT and can be dequeued after ET in a non-work-conserving manner. It operates at a constant interval that depends on the packet size and the service rate.

N-SCORE manages per-flow state to calculate ET and FT at the entrance node. However, core nodes do not maintain state to accommodate large-scale networks. As a result, N-SCORE calculates and applies ET and FT differently at the entrance node and subsequent core nodes.

Whenever a packet arrives, the entrance node calculates its ET and FT based on the managed per-flow state, updates the state using the calculated FT, and appends ET and FT as metadata to the packet header. Subsequent core nodes retrieve ET and FT from the metadata without maintaining state separately. At the same time, they calculate new ET and FT for the next node and update the metadata accordingly.

6.1. Operational Procedure in Entrance Node

The entrance node manages the per-flow state, including the FT of the previous packet, F(pâ\210\2221), and the service rate assigned to the flow, r(p). When a packet arrives at the entrance node, its ET, E(p), is determined as max{F(pâ\210\2221), A(p)}. The entrance node compares each packet's arrival time, A(p), with the managed F(pâ\210\2221) and sets the later time as E(p). The FT of the arriving packet, F(p), is calculated as E(p)+L(p)/r(p), and the FT of the previous packet is updated with the newly obtained F(p). Packets are stored in a priority queue in ascending order of E(p) and F(p) and can be dequeued after E(p) in a non-work-conserving manner.

When the packet arrival interval is greater than the service rate, as seen with the first and second packets in Figure 2, the arrival times of these packets at node 1, A1(1) and A1(2), are later than the FT of

the previous packet managed by the entrance node, F1(0) and F1 (1), respectively. Therefore, the ET of the first and second packets at node 1, E1(1) and E1(2), are set as A1(1) and A1(2), respectively. In this case, the service period is (A(p), A(p)+L(p)/r(p)], which matches the service period of C-SCORE.

However, when the packet arrival interval is smaller than the service rate, as seen with the third and fourth packets in Figure 2, the arrival times of these packets at node 1, A1(3) and A1(4), are earlier than the FT of the previous packet managed by the entrance node, F1(2) and F1(3), respectively. Consequently, the ET of the third and fourth packets at node 1, E1(3) and E1(4), are set as F1(2) and F1(3), respectively. In this case, unlike C-SCOREâ\200\231s service period of (A(p), F(pâ\210\2221) + L(p)/r(p)], the N-SCORE service period is (F(pâ\210\2221), F(pâ\210\2221) + L(p)/r(p)]. N-SCORE regulates packet transmission based on the service rate, ensuring a deterministic and non-overlapping service period for all packets.

A packet is dequeued after E(p), and before leaving, the entrance node marks metadata in the packet header, including L(p)/r(p), as well as the ET and FT for the next node. The subsequent core nodes then use this metadata to determine their ET and FT.

6.2. Operational Procedure in Core Node

When the ET and FT of a packet are determined at the entrance node, the ET and FT of all subsequent nodes are determined based on the previous node's ET and FT as follows:

Eligible Time for the next node:

$$E_{h+1}(p) = E_h(p) + d_h(p) \quad (7)$$

Finish Time for the next node:

$$F_{h+1}(p) = F_h(p) + d_h(p) \quad (8)$$

Here, dh(p) represents the maximum delay within node h, which is calculated as:

$$d_h(p) = L(p)/r(p) + L_{hmax}/R_h \quad (9)$$

The term Lhmax/Rh accounts for delay factors at node h, where Lhmax is the max packet length at node h across all flows transmitted through the observed output port, and Rh is the link capacity of node h.

The entrance node delivers the metadata, including L(p)/r(p), ET, and FT, through the packet header. Subsequent core nodes obtain their ET and FT from the metadata without per-flow state management. Based on its delay factors and L(p)/r(p) value in the metadata, each core node computes dh(p), determines the ET and FT for the next node, and updates the metadata accordingly.

Packets are stored in a priority queue in ascending order of E(p) and F(p), as derived from the metadata, and can be dequeued after E(p) in a non-work conserving manner.

7. Capability Analysis

Based on the draft of the taxonomy, latency-bound solutions are classified according to functional characteristics such as

* periodicity (periodic, non-periodic)
* network synchronization (phase and frequency synchronous, asynchronous)
* traffic granularity (flow level, flow aggregate level, class level)
* work conserving (work conserving, non-work conserving)
* target transmission time (in-time, on-time)
* service order (rate-based, time-based, arrival-based, priority-based)

The solutions proposed in DetNet and TSN can be broadly classified into seven types according to their functional characteristics:

1. Non-periodic/asynchronous/flow level/work conserving/in-time/ rate-based solution (C-SCORE)
2. Non-periodic/asynchronous/flow level/non-work conserving/in-time/ rate-based solution (ATS)
3. Non-periodic/asynchronous/class level/work conserving/in-time/ time-based solution (In-time EDF)
4. Non-periodic/asynchronous/class level/non-work conserving/on-time/time-based solution (On-time EDF)
5. Non-periodic/asynchrounous/flow aggregate level/non-work conserving/on-time/time-based solution (PIFO based on-time)

6. Periodic/phase synchronous/class level/non-work conserving/on-time/time-based solution (TAS, CQF)

7. Periodic/frequency synchronous/class level/non-work conserving/on-time/time-based solution (TQF, CSQF, TCQF, ECQF)

The current solutions that provide on-time characteristics are all solutions with class-level or flow aggregate level traffic granularity characteristics and time-based service order characteristics. This draft suggests a new type of solution that has not been proposed before:

8. Non-periodic/asynchronous/flow level/non-work conserving/on-time/rate-based solution (N-SCORE)

The proposed solution is an on-time solution with rate-based service order characteristic that can handle a large number of dynamic flows with simple admission control. Additionally, it has flow-level traffic granularity characteristics that can minimize the effects of other flows' bursts.

8. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

9. Security Considerations

TBD

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

Authors' Addresses

Yeoncheol Ryoo
ETRI
Email: dbduscjf@etri.re.kr

Jinoo Joung
Sangmyung University
Email: jjoung@smu.ac.kr

DetNet Working Group
Internet-Draft
Intended status: Standards Track
Expires: 30 August 2025

Y. Ryoo
ETRI

26 February 2025

On-time Forwarding with Push-In First-Out (PIFO) queue
draft-ryoo-detnet-ontime-forwarding-02

Abstract

This document describes operations of data plane and controller plane for Deterministic Networking (DetNet) to forward packets to meet minimum and maximum end-to-end latency requirements, while utilizing Push-In First-Out (PIFO) queue.

According to the solution described in this document, forwarding nodes do not need to maintain flow states or to be time-synchronized with each other.

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 30 August 2025.

Copyright Notice

Table of Contents

1. Introduction

Deterministic Networking (DetNet) whose architecture is defined in [RFC8655] provides the capability to carry specified unicast or multicast flows with extremely low packet loss rates and bounded end-to-end latency.

On-time forwarding is a critical feature of deterministic networks, especially of networks dealing with industrial process control signaling. The on-time forwarding is characterized as packets belonging to a flow are delivered within minimum end-to-end latency (MinLatency) and maximum end-to-end latency (MaxLatency) requirements for the flow. The difference between MaxLatency and MinLatency is the end-to-end latency variation, which becomes smaller as the requirement for on-time delivery precision becomes stricter. When MinLatency does not require to be guaranteed, it can be viewed as in-time forwarding.

This document describes operations of data plane and controller plane for DetNet to forward packets to meet minimum and maximum end-to-end latency requirements, while utilizing Push-In First-Out (PIFO) queue. Given MinLatency and MaxLatency requirements for a flow and non-queuing delays and available buffer resources on the path selected for the flow, the controller calculates lower and upper node delay bounds for each node on the path. When a packet arrives at a node, the node computes minimum departure time, nominal departure time, and maximum departure time for the packet based on the lower and upper node delay bounds calculated by the controller for the node. Using the PIFO queue, the packets are arranged in the ascending order of their nominal departure times in the PIFO queue and forwarded between their minimum and maximum departure times.

2. Terminology

2.1. Symbols Used in This Document

E2E_F	end-to-end fixed delay
E2E_VL	end-to-end variable delay lower bound
E2E_VU	end-to-end variable delay upper bound
MaxLatency	maximum end-to-end latency that must be guaranteed
MinLatency	minimum end-to-end latency that must be guaranteed
N_L	node delay lower bound
N_U	node delay upper bound
R_L	remaining end-to-end latency lower bound
R_U	remaining end-to-end latency upper bound

2.2. Abbreviations

3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

4. Temporal Model

This document separates end-to-end latency into two components: end-to-end variable delay and end-to-end fixed delay. The end-to-end variable delay is the sum of variable delays occurring in nodes and links on the path, and has its upper and lower bounds, which are denoted by E2E_VU and E2E_VL, respectively. Using the terms defined in [RFC9320], one obvious example of a variable delay is a queuing delay. Other delays, such as output delay, link delay, and

processing delay, can be classified as variable delays depending on implementation. On the other hand, the end-to-end fixed delay, denoted by E2E_F, is the sum of fixed delays occurring in links and nodes on the path. Some or all of the delays except the queuing delay can be included in the E2E_F depending on how the nodes and links are implemented. When an implementation can provide a fixed value for any non-queuing delay, that delay is considered a fixed delay in this document. An example of a fixed delay is the first-bit-out to first-bit-in delay of the link delay [RFC9320] unless the link is formed virtually. When a flow consists of packets of a constant size, the first-bit-in to last-bit-in delay of the link delay [RFC9320] also becomes a fixed delay. In this document, we assume that the first-bit-out to first-bit-in delay, which is commonly called link propagation delay, is classified as a fixed delay that depends on length of a link.

When a flow is requested, the non-queuing delays are known to a controller by considering network topology, port speeds, link lengths, maximum and minimum processing and output delays of nodes, and maximum and minimum packet sizes of the flow.

In order to guarantee MaxLatency and MinLatency, the sum of E2E_F and E2E_VU MUST be less than or equal to MaxLatency, and the sum of E2E_F and E2E_VL MUST be greater than or equal to MinLatency. Figure 1 shows the relationship among the aforementioned end-to-end parameters.

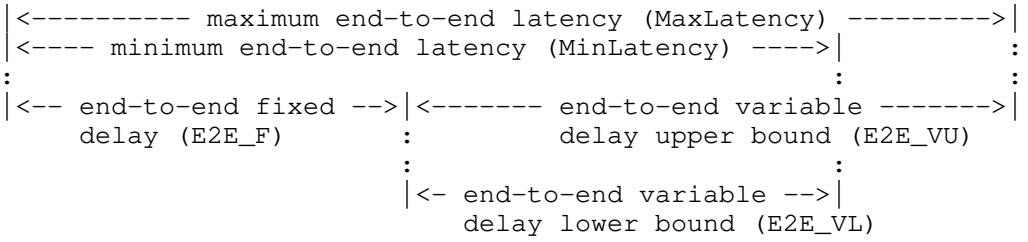

Figure 1: Relationship among end-to-end latency parameters

For the data plane operation described in the document, E2E_VU and E2E_VL are divided into all the nodes on the path, and the controller assigns a node delay upper bound (N_U) and a node delay lower bound (N_L) to each node. N_U and N_L are upper and lower bounds of the time a packet can reside in a node, and their values may be different for each node. For the sake of brevity, we omit the index for a node in this document. How the controller determines the values of N_U and N_L is described in Section 6.

Once N_U and N_L are given, a node performs the data plane operation as described in Section 5.

5. Data Plane Operation

As mentioned in the previous section, N_L and N_U are assumed to be set in each node on the path. The values of (MinLatency-E2E_F) and (MaxLatency-E2E_F) are also assumed to be known at the first node on the path. In addition to the normal DetNet encapsulation, such as DetNet control word, service label, and forwarding labels in case of MPLS, this document assumes that fields containing upper and lower bounds of remaining end-to-end latency, called R_L and R_U, are available. A node is assumed to measure a residence time, which is defined as the time a packet resides in the node. To be more precise, the residence time is the time duration between the time that the last bit of a packet comes in and the time that the last bit of the packet leaves the node.

When a packet arrives at the first node on the path, the node performs the queuing operation as described in Section 5.1 based on N_L and N_U values assigned to the first node. When the packet departs from the first node, R_L and R_U fields are set by subtracting its residence time from (MinLatency-E2E_F) and (MaxLatency-E2E_F), respectively.

Each node except the first and last nodes on the path performs the queuing operation as described in Section 5.1 based on N_L and N_U values assigned to the node, and updates R_L and R_U fields by subtracting its residence time from received R_L and R_U values. If the resulting value of R_L is negative, then the R_L field is updated with zero.

The last node also performs the queuing operation as described in Section 5.1, but uses R_L and R_U received from its previous node instead of N_L and N_U values assigned to the last node. If R_U is greater than N_U of the last node, N_U is used.

5.1. Queuing Operation

When a packet is processed within a node, it can experience variable delays, which are generally categorized into three types of delay: processing delay, queuing delay, and output delay. N_U and N_L represent the total variable delay that the node must guarantee. Processing delay and output delay vary depending on the packet size, etc, but these delays are hard to adjust, so the queuing delay must be adjusted to guarantee N_U and N_L. Therefore, when a packet is received by a node, the expected output delay that may be required based on the packet size is subtracted from N_U and N_L, and the remaining values are determined as the packet's minimum, and maximum departure times. Since the output delay can also have a variable value, it is calculated as follows. When a packet arrives at time t, a minimum departure time, which is defined as t plus N_L minus del_min, and a maximum departure time, which is defined as t plus N_U minus del_max, are calculated. del_min and del_max are defined as the minimum and maximum times it takes from the time a packet leaves the queue until it completely leaves the node. del_min or del_max can be calculated with the size of the packet, port speed and minimum or maximum output delay, respectively. In addition, a nominal departure time, which is defined as the midpoint between the minimum departure time and maximum departure time, is calculated. The difference between (N_U - del_max) and (N_L - del_min) is called forwarding budget. Figure 2 shows the relationship among the minimum, nominal, and maximum departure times.

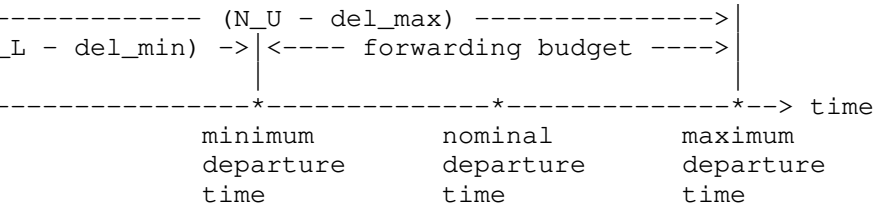

Figure 2: Relationship among the minimum, nominal, and maximum departure times

After calculating the minimum, nominal, and maximum departure times and performing necessary actions for packet forwarding, the packet is placed in the PIFO queue, where packets are arranged in the ascending order of their nominal departure times. When the minimum departure time of the packet in the head of queue (HoQ) has reached or passed current time, the packet is dequeued. Since the minimum and maximum departure times that a packet can stay in the queue are determined by

N_U and N_L, if the time taken from when the packet is received until it is processed and queued is long, the actual queuing time will automatically decrease. Conversely, if the time taken until it is queued is short, the queuing time will increase.

An example of the queuing operation is shown in Figure 3. Let us consider three incoming packets belonging to three different flows. The values of N_L and N_U are set to 1ms and 3 ms for the first flow, 0.34ms and 2ms for the second flow, and 0.3ms and 0.5ms for the third flow, respectively. To simplify the example, we assume del_min and del_max are zero.

* Assume that the first packet, P1, arrives at 0.2ms. Then, the minimum, nominal, and maximum departure times of P1 are calculated as 1.2ms, 2.2ms, and 3.2ms, respectively. P1 is placed at the HoQ and cannot leave the queue before 1.2ms.

* When the second packet, P2, arrives at 0.4ms, the minimum, nominal, and maximum times of P2 are determined as 0.74ms, 1.57ms, and 2.4ms. Since the nominal departure time of P2 is smaller than that of P1, P2 is placed at the HoQ and is scheduled to leave the queue at 0.74ms.

* The third packet, P3, is assumed to arrives at 0.6ms and its minimum, nominal, and maximum departure times are calculated as 0.9ms, 1.0ms, and 1.1ms, respectively. Since the nominal departure time of P3 is smaller than that of P2, P3 is placed at the HoQ and is followed by P2 and P1.

* At 0.9ms, P3 leaves the queue as its minimum departure time is 0.9ms. Following P3, P2 immediately leaves the queue as its minimum departure time (0.74ms) has passed.

* At 1.2ms, P1 is dequeued as its minimum departure time is 1.2ms.

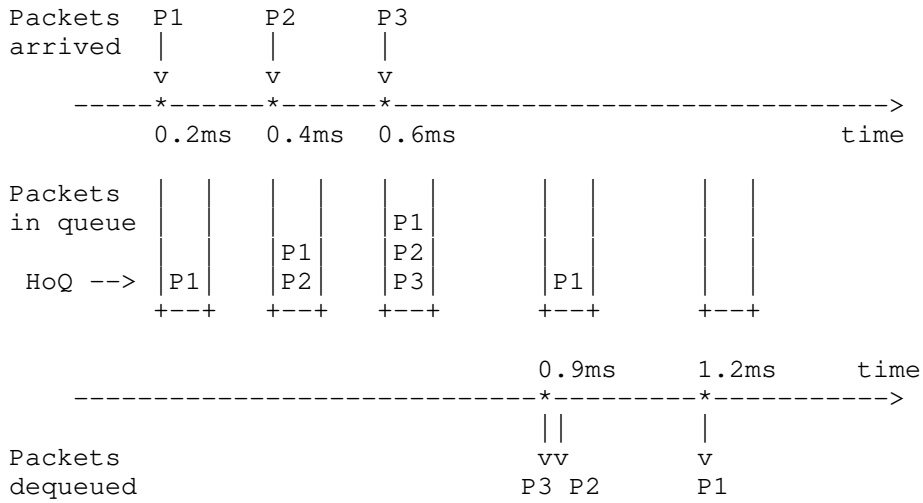

Figure 3: Example of queuing using PIFO queue

In this queuing operation, if packets with nominal departure times smaller than the nominal departure time of the HoQ packet continue to arrive, the packet with a small forwarding budget may exceed its maximum departure time. Therefore, the forwarding budget MUST be set to be larger than the time required to transmit any preceding packets of all the flows at the speed of the output port. This requirement of the forwarding budget needs to be confirmed through admission control in the controller plane when the flow is set up.

6. Controller Plane Operation

A controller collects network topology, PIFO queue resource, and various delay-related information such as port speed, link length, maximum and minimum processing and output delays of nodes, and maximum and minimum packet sizes of the flow, etc.

If a new DetNet flow is requested, the controller selects a path that satisfies the following conditions:

1. The E2E_F of the path MUST NOT exceed the MaxLatency required for the flow.

2. The sum of the available buffer resources of all nodes on the path MUST be large enough to provide a delay greater than E2E_VL minus minimum end-to-end variable non-queuing delay.

3. The available buffer resource of each node on the path MUST be large enough to provide a delay equal to N_U minus minimum node variable non-queuing delay.

4. The forwarding budget of each node MUST be larger than the time required to transmit any preceding packets of all the flows at the speed of the output port.

The first condition can be easily checked with the fixed delay values collected by the controller.

The second condition represents the minimum end-to-end queuing delay which is the minimum packet queuing time that nodes along the end-to-end path must guarantee. Since the maximum queuing delay can be obtained by dividing the buffer size by the service rate, whether condition 2 is satisfied can be checked as â\210\221[h=0,H] {buffer sizes} / the service rate >= E2E_VL - minimum e2e variable non-queuing delay.

In the second condition, the minimum end-to-end variable non-queuing delay is defined as the sum of lower bounds of variable delays except queuing delays occurring in nodes and links on the path, and the controller is assumed to be able to calculate from the information collected from the network. Likewise, in the third condition, the minimum node variable non-queuing delay is defined as the sum of lower bounds of variable delays except a queuing delay in a node, and the controller is assumed to be able to calculate from the information collected from the node.

In order to check the third and fourth conditions, N_L and N_U for each node need to be determined. There can be various ways to determine the values of N_L and N_U. In the following, we describe how both N_U and N_L can be obtained as one of the possible ways.

Considering the fact that the last node is the node that can take final actions to ensure the E2E_VL and E2E_VU for packets requiring on-time delivery, the value of N_U of the last node MUST be determined first. It is RECOMMENDED to set the value of N_U of the last node as large as possible as long as the buffer resource of the last node allows for the flow. N_U is computed as the maximum queuing delay plus the minimum node variable non-queuing delay. Since the maximum queuing delay is determined by the buffer size allocated to a flow, If a policy is defined, such as setting a maximum buffer size per flow to prevent a single flow from occupying all the buffer resources of the node, the maximum queuing delay is calculated based on the maximum buffer size specified by the policy. If no such policy exists, the operator considers the buffer size allocated per flow and determines N_U to be as large as possible without exceeding the total buffer size of the node. Then, the

remaining value after subtracting N_U of the last node from E2E_VL is divided into all other nodes. The value divided into each node is used as N_L for the node. The N_L of the last node is set to the time required to transmit any preceding packets of all the flows at the speed of the output port of the last node.

The value of N_U of each node except the last node is determined by dividing the remaining value after subtracting N_L of the last node from E2E_VU into all nodes except the last node on the path. Figure 4 shows the relationship between the variable delay of end-to-end and nodes

If the available buffer resource of each node on the path can support the value of N_U minus minimum node variable non-queuing delay, the third condition is satisfied. The fourth condition can be checked with N_U and N_L.

Once a path satisfying the aforementioned conditions is selected, the values of N_L and N_U are set to all nodes, and each node performs the operation described in Section 5 in the data plane. And, the buffer resources associated with N_U become unavailable for flows requested later.

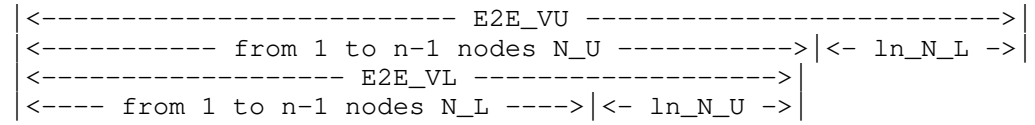

Figure 4: Relationship between the variable delay of end-to-end and nodes

7. Capability Analysis

The data and controller plane operations described in this document have the following characteristics for the requirements described in [I-D.ietf-detnet-scaling-requirements]. The item numbers below correspond to the numbers of the technical requirements in Section 3 of [I-D.ietf-detnet-scaling-requirements].

1. The solution described in this document does not require time synchronization. However, the solution measures the residence time and passes the remaining end-to-end latency values to the next node. As a specific delay value seen by all nodes must be the same amount, frequency synchronization is necessary.

2. The large single-hop propagation delay is supported. The solution describe in this document does not impose any limits on the amount of propagation delay.

3. Accommodation of the higher link speed is supported. It is considered possible to implement a PIFO queue supporting speeds of 100 Gbps or more.

4. The solution described in this document is scalable to the large number of flows as it does not require to maintain flow states in a node.

5. The solution described in this document is robust against node and link failures and topology changes, as the PREOF function can be applied.

6. Since the solution described in this document provides on-time forwarding while complying with the forwarding budget at all nodes, flow fluctuation inherently does not occur.

7. Since each node operates independently and the operation of the controller does not require any greater burden than existing typical network control, there are no scalability issues regarding the number of hops.

8. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

9. Security Considerations

TBD

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[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/info/rfc9320>.

10.2. Informative References

[I-D.ietf-detnet-scaling-requirements] Liu, P., Li, Y., Eckert, T. T., Xiong, Q., Ryoo, J., zhushiyin, and X. Geng, "Requirements for Scaling Deterministic Networks", Work in Progress, Internet-Draft, draft-ietf-detnet-scaling-requirements-07, 20 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-scaling-requirements-07>.

Author's Address

Yeoncheol Ryoo
ETRI
Email: dbduscjf@etri.re.kr