DetNet Working Group
Internet-Draft
Intended status: Standards Track
Expires: 24 August 2026

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

20 February 2026

Latency Guarantee with Stateless Fair Queuing
draft-joung-detnet-stateless-fair-queuing-07

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 the FT by adding the delay factor, which is a function 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. Furthermore, this document specifies an approximation of stateless fair queuing implemented via a strict priority (SP) scheduler. This approach maintains a guaranteed end-to-end (E2E) latency bound.

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/.

This Internet-Draft will expire on 24 August 2026.

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. The objective of this document is to specify a scheduling mechanism that can isolate flows from each other. 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 provides 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.

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, without per-flow state, by adding a delay factor that is a function of parameters of the node and the flow. This framework is called work conserving stateless core fair queuing (C-SCORE) [C-SCORE], which is also specified in [Y.3129] and [Y.3148]. 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.

This document specifies the protocol and implementation details for realizing C-SCORE. It also specifies the operational procedure based on these details.

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

BE: Best Effort

C-SCORE: Work Conserving Stateless Core Fair Queuing

DetNet: Deterministic Networking

E2E: End to End

FQ: Fair Queuing

FT: Finish Time

GPS: Generalized Processor Sharing

HoQ: Head of queue

FIFO: First-In First-Out

MNA: MPLS Network Actions

NAS: Network Action Sub-Stack

PIFO: Push-In First-Out

PRPS: Packetized Rate Proportional Servers

PSD: Post-Stack Data

RSpec: Requested Specifications

TLV: Type-Length-Value

TSpec: Traffic Specifications

VC: Virtual Clock

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 or equivalently DetNet traffic requires guarantee on latency upper bounds. All the other traffic is considered to be the low priority or Best Effort (BE) 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 (6) 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 defined as having arrived or been serviced when its final bit has been received by or transmitted from the node, respectively.

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

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

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.

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.

$$F0(p) = \max\{F0(p-1), A0(p)\}+L(p)/r. \quad (3)$$

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.

$$Fh(p) = F(h-1)(p) + d(h-1)(p). \quad (4)$$

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. 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.

This framework is called the work conserving stateless core fair queuing (C-SCORE) [C-SCORE], [Y.3129], [Y.3148], 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.

$$dh(p) = Lh/Rh + L/r + delta\_h(p), \qquad (5)$$

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, L is the maximum packet length of the flow, r is the service rate of the flow, and delta_h(p) is the time difference function between node h and h+1.

The time difference function of packet p is defined as the service completion time at node h and the arrival time at node h+1 of the packet. It includes the clock discrepancy and the propagation delay between nodes. Note that this function is relatively stable over packets, thus should be approximated as a constant value delta_h.

When dh(p) is decided by (5), then it is proved that

$$Dh(p) <= (B-L)/r + sum\_(j=0)^h\{Lj/Rj + L/r\} + sum\_(j=0)^(h-1)\{delta\_j(p)\}, \qquad (6)$$

where Dh(p) is the latency experienced by p from the arrival at the node 0 to the departure from node h [KAUR], [C-SCORE]. B is the maximum burst size of the flow under observation that p belongs to. The term sum_(j=0)^(h-1){delta_j(p)} includes the clock discrepancies and the propagation delays between the nodes. Considering that the clock discrepancies can be neglected in an absolute time, this term actually represents the E2E propagation delay.

Note that the latency bound in (6) 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

The necessary metadata to be carried by a packet are Fh(p) and L/r. Fh(p) is dynamic and needs to be updated every hop, in accordance with (4). L/r is a static value for a flow. Note that these metadata both have the unit of time.

As a packet arrives at a core node h, it carries metadata Fh(p) and L/r. Fh(p) is pre-calculated at node h-1 with d(h-1)(p), which is a function of node h-1. dh(p) can be obtained by the summation of the metadata L/r, the node specific parameters Lh/Rh and delta_h. L/r should be kept in a packet as metadata, to be able to calculate dh(p), in accordance with (5). Lh/Rh and delta_h values should be maintained by the node h.

6.3.2. Header format

6.3.2.1. IPv6 header format

The IPv6 Hop-by-Hop (HbH) Options Header [RFC8200] should be used for carrying the metadata. It is a specific type of Extension Header designed for information that must be examined and processed by all the transit nodes along a packet’s delivery path. This header is identified by a Next Header value of 0 in the IPv6 main header. The HbH Options header consists of a sequence of variable-length options. These options are encoded in a Type-Length-Value (TLV) format. In this transformation, the metadata (Finish Time and L/r) are stored in the Option Data fields of two separate Options within a single HbH Options header. Length of the HbH Options header should be specified by the Hdr Ext Len field. Option Type and Option Data Len fields should specify the C-SCORE metadata identifier and the length of the metadata, which are to be determined.

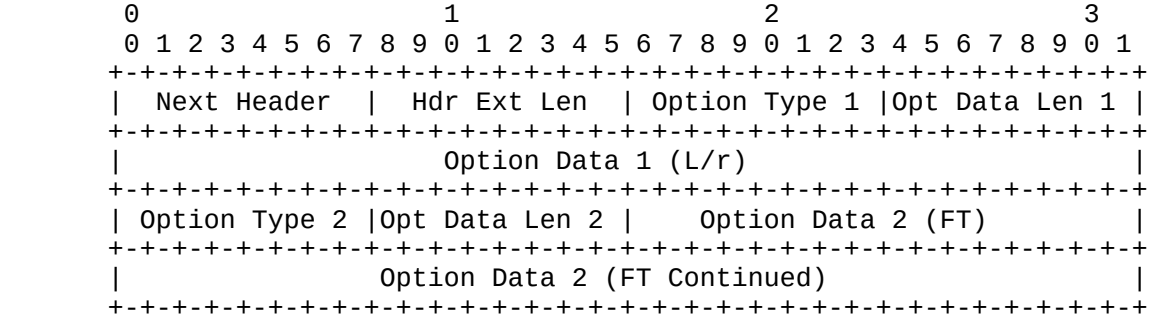

Figure 1: IPv6 HbH Options Header example for two metadata for C-SCORE

The following is the operational procedure in a transit node. The hardware parser sees Next Header 0 in the main IPv6 header and directs the packet to the metadata management function. This functional entity scans for the Option Type corresponding to C-SCORE. The Finish Time (48-bit, TBD) and L/r (32bit, TBD) are extracted and processed. The new FT is calculated. Since the offset is fixed relative to the start of the HbH Options header, the functional entity performs a single cycle write to update the metadata field before recalculating any necessary checksums (if applicable) and forwarding.

6.3.2.2. MPLS label format

[I-D.ietf-mpls-mna-detnet] specifies formats and mechanisms for using MPLS Network Actions (MNA) to support DetNet services, including bounded latency, low loss and in-order delivery. It specifies three information elements of DetNet packets, which are Flow identifier (Flow-ID), Sequence information (SeqNum), Latency information (LatencyInfo). The C-SCORE metadata Fh(p) and L/r are the Latency Information, according to this specification.

Two approaches for carrying information elements are specified: In-Stack and Post-Stack MNAs. With In-Stack MNA, the DetNet-specific information is embedded directly within the MPLS label stack, as part of a Network Action Sub-stack (NAS). The information elements reside before the Bottom of Stack (BOS) bit. It uses a Network Action Indicator (NAI) to signal that the subsequent labels in the sub-stack are actually ancillary data (Flow-ID, etc.) rather than traditional switching labels.

With Post-Stack MNA, the DetNet-specific information is carried after the label stack. The data resides between the BOS bit and the start of the user payload. An indicator within the label stack (the NAI) points to the presence of Ancillary Data located immediately after the stack. Post-stack data is better suited for large or variable-sized data that would otherwise make the label stack prohibitively deep.

This document follows the Post-Stack encoding approach, but the In-Stack approach is not excluded. In the Post-Stack approach, the MNA sub-stack is usually placed immediately after the bottom of the MPLS label stack. This allows for large amounts of ancillary data to be carried without making the label stack excessively deep.

The Post-Stack MNA solution contains two components:

1) Post-Stack MPLS Header Presence Bit carried in In-Stack MNA Sub-Stack

2) Post-Stack MPLS Header that includes Post-Stack MPLS Header Type (PSMHT) and Post-Stack Network Actions (PSNA)

Bit 20 in Label Stack Entry (LSE) Format B carried in the In-Stack NAS is defined as the P bit to indicate the presence of the Post-Stack MPLS Header in the packet after the BOS bit. [I-D.ietf-mpls-mna-ps-hdr] LSE Format B refers to a specialized structure for a LSE that carries ancillary data instead of a traditional switching label.

PSMHT can be located immediately following the BOS label, which includes PS-HDR-LEN and Version/Type. PS-HDR-LEN specifies the total length of the post-stack metadata. Version/Type identifies the MNA-POST-STACK-HDR. The metadata for C-SCORE are Finish Time and L/r. The number of bits required for these metadata are for further study and to be specified. FT and L/r are carried across two separate PSNAs to support hop-by-hop scheduling. If FT is of 48 bit length, then the PSNA for FT should have Post-Stack Network Action Length (PS-NAL) = 1, as in Figure 2.

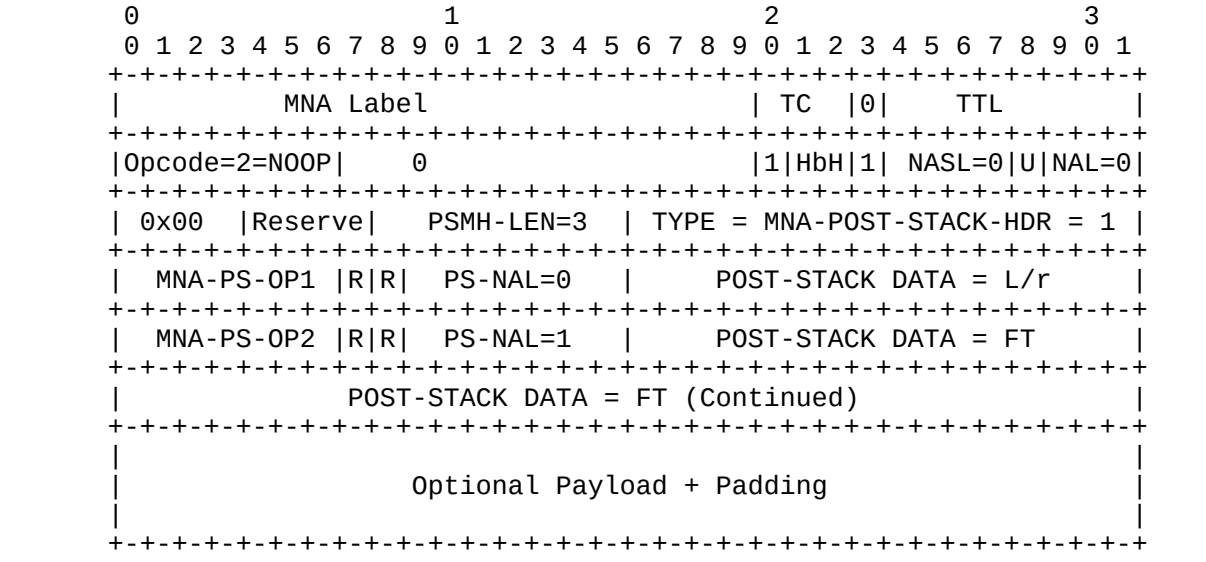

Figure 2: Post Stack MNA Sub-Stack example with two PSNAs for two metadata for C-SCORE

Figure 2 is an example where the Post-Stack MNA Sub-Stack encodes two different PSNAs for two metadata. Their details are as follows:

- The offset of the Hop-By-Hop scoped PSNA is 0.

- PSMH-LEN=3: This is the total length of the Post-Stack MPLS Header (PSMH).

- MNA-PS-OP1: Post-Stack MNA Opcode (TBD) for L/r.

- PS-NAL=0: PSNA does not contain any additional data.

- MNA-PS-OP2: Post-Stack MNA Opcode (TBD) for Finish Time.

- PS-NAL=1: PSNA contains 1 additional 4-octet Ancillary Data.

Note that C-SCORE does not require Flow-ID to be carried in the packet. This is because of its stateless nature.

6.3.3. Admission control for latency guarantee

The following is a recommended procedure for admission of a flow in the C-SCORE framework.

1) An application requests a flow with T-Spec (Traffic Specification) and R-Spec (Requested Specification). T-Spec includes service rate r, maximum packet size L, and maximum burst size B. R-Spec includes the E2E latency and jitter bounds.

2) The entrance node sends a PATH message toward the egress. This message contains the T-Spec and R-Spec.

3) The entrance node acts as the admission controller. It maintains per-flow state and performs the initial shaping, if necessary.

4) Core nodes check if the requested service rate r is within the aggregate bandwidth threshold of the outgoing interface.

5) Each core node adds its per-hop max latency value (L/r + Lh/Rh) to a field in the PATH message. This allows the egress node to calculate the E2E latency bound.

6) The egress node receives the PATH message. If the E2E latency bound meets the application’s requirement, it generates a RESV message.

7) The RESV message travels back to the entrance node. The core nodes in the path confirms the admission of the flow.

8) If the RESV message reaches the entrance node, the flow is admitted.

In a C-SCORE architecture utilizing the RSVP-like admission process described above, it is not only possible but highly efficient to find out and negotiate the optimal available service rate for a flow. The admission process can probe the network to determine the maximum supportable rate without violating the deterministic latency bounds of existing flows. The discovery of the available rate occurs during the PATH message traversal. The following is an alternative admission process, which identifies the tightest capacity link in the path.

1) An application requests a flow with T-Spec (Traffic Specification) and R-Spec (Requested Specification).

2 The entrance node sends a PATH message with a Desired Rate (r_desired) and a Minimum Acceptable Rate (r_min), along with T-Spec and R-Spec.

3) Each core node on the path calculates its residual capacity (r_avail). This is the total capacity of the link minus the sum of the service rates (r) of all already admitted flows.

4) As the PATH message moves hop-by-hop, it maintains a field called Path-Available-Rate (r_path). At each hop h, the router performs: r_path = min(r_path, r_avail_h), while the subscript h denotes the node.

5) By the time the PATH message reaches the egress, r_path represents the maximum service rate the entire end-to-end path can support at that specific moment.

6) The optimal rate is not always the highest possible rate. The egress node can select the optimal rate, which can be less than r_path.

7) Once the optimal service rate (r = r_opt) is determined, the egress sends the RESV message back to the entrance node, carrying the r_opt value.

8) The core nodes in the path confirms the r_opt value for the flow.

9) If the RESV message reaches the entrance node, the flow is admitted.

10) The entrance node commits this rate to its per-flow state.

The optimal rate should be between r_path and r_min. The egress node can select the optimal rate based on the following criteria:

- Latency requirements: Higher service rates result in smaller Finish Time (FT) increments, reducing the per-hop queuing delay.

- Buffer constraints: The rate must be balanced with the max burst (B) parameter to ensure the core nodes' buffers do not overflow. Note that the burst accumulates linearly with the service rate.

- Network availability: The egress may choose a rate lower than r_path to leave room for other flows that can join later.

Since the core nodes only need to know the final committed rate r in the packet header, the complex discovery logic is restricted to the control plane and edge nodes. Large-scale networks can use this mechanism to perform the aggregate rate discovery, where the r_path represents the available capacity for an entire bundle of flows of the same path. By using this approach, the admission process also effectively maps the current congestion state of the network onto a single service rate parameter.

6.3.4. 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).

6.3.5. 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.

6.3.6. 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.

6.3.7. Compensation of time difference between nodes

There are time differences between nodes, including the clock discrepancies and 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,

$$delta\_h(p) = A(h+1)(p) - Ch(p),$$

where delta_h(p) is the time difference between node h and h+1, and Ch(p) is the service completion time measured at node h, 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 delta_h(p) can be simplified to a constant value, delta_h. In this case the delay factor in (4) can be modified to be

$$dh(p) = Lh/Rh + L/r + delta\_h.$$

The time difference delta_h may be updated only once in a while.

The protocol for obtaining the departure time from node h, Ch(p), at node h+1 will be elaborated in a later version of this draft. The Ch(p) information can be obtained on demand, or be reported periodically. The message format will follow that of Network Time Protocol (NTP), but the procedure will be simpler. The message itself can be standalone like in NTP, or can be piggybacked on a data packet.

Note that the time difference between non-adjacent nodes can also be obtained similarly. This feature is useful when there are non-compliant nodes in between. In this case, however, the variable queuing delay from the non-compliant nodes should be taken into account. One possible solution is to sample the time difference values over an enough interval, and take the maximum value.

6.4. Characteristics

6.4.1. Taxonomy

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.ietf-detnet-dataplane-taxonomy].

[I-D.ietf-detnet-dataplane-taxonomy] also defines seven suitable categories for deterministic networking. A category is defined to be a set of solutions that is put together by one or more criteria, where a criterion is a principle or standard by which a solution can be judged or decided to be put into a certain category.

C-SCORE belongs to the "flow level rate based unbounded category", which is one of the seven suitable categories, according to this categorization.

6.4.2. Strengths

C-SCORE’s per hop latency dominant factor is the maximum packet length divided by the service rate of the flow. This is 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 in real situations. 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. Approximate C-SCORE via strict priority schedulers

7.1. General description

C-SCORE requires a priority queue that sorts the packets in a queue, in accordance with their finish times. This may restrict the overall maximum throughput of a system, when compared to the strict priority (SP) schedulers used in the current practices of switching nodes. SP schedulers are usually composed of 8 to 32 queues, and schedule the packets of higher priority queue first whenever they are present. The packets in the same queue are served on a first in first out (FIFO) basis. It is common to find the SP schedulers in hardware chips for current switching nodes.

It would be desirable if C-SCORE can be implemented with such an SP scheduler. In this section, the architecture and algorithms for the Approximate C-SCORE with rotating SP schedulers are specified. The E2E latency bound of the network of the approximate C-SCOREs is also specified.

7.2. Transit node architecture

The architecture of an approximate C-SCORE transit node, in which the SP scheduler with a limited number of queues behaves as an approximate priority queue, is specified in this section.

The input port module classifies the deterministic flows and best effort (BE) flows. The deterministic flows are put into one of N queues that work as rotating SP scheduler queues. If the queue k, 0 <= k <= N-1, is the highest priority queue at time t, then the queue (k+N-1)(mod N) has the lowest priority at t. BE flows are put into its own queue in the output port module. BE queue is served only when there is no packet in queues 0 to N-1.

Assume that the time is divided into fixed-length slots. Let us say that a slot is allocated to a certain queue. Further, let T_i denote the terminal boundary of Slot i. An arriving packet p is assigned to Slot i if its finish time, F(p), falls within the interval (T_(i-1),T_i]. This mapping corresponds to a discrete set of hardware queues where packets are buffered and processed according to a First-In-First-Out (FIFO) discipline. The system utilizes a strict priority (SP) scheduler to arbitrate across these queues, granting precedence to those representing the earliest finish-time. The scheduler operates in a work-conserving manner, providing service at line rate whenever the system is backlogged.

7.3. Algorithms

To provide differentiated quality-of-service, the per-hop latency must be modulated according to the specific service rates of traversing flows. By reducing the scheduling granularity, or slot length (S), the system can provide tiered latency bounds based on the number of slots occupied by a flow’s maximum virtual service interval (L/r). Specifically, for a flow f where (n-1)S < L/r <= nS, the per-hop latency is bounded by (n+1)S. This mechanism ensures fairness by granting lower latency to flows with higher service rates (smaller L/r), thereby aligning temporal performance with bandwidth allocation. The scheduling granularity, or slot duration S, is defined such that S >= min_p[L(p)/r(p)]. While a finer granularity generally minimizes end-to-end latency, reducing S below this lower bound, which is determined by the minimum packet transmission time relative to the flow rate, yields no additional scheduling benefit. In practical implementations, N hardware queues are allocated and managed in a cyclic manner to accommodate these slots. Consequently, the selection of S involves a fundamental trade-off: a coarser slot duration allows for a reduced number of physical queues, N, at the expense of decreased scheduling precision.

In an idealized preemptive system, the scheduler ensures that all packets mapped to a specific slot are fully serviced before its terminal boundary, T_i. Consequently, the service interval for any packet p is strictly contained within its assigned slot. However, in a realistic non-preemptive environment, the completion time is extended by a factor of L_h/R_h, where L_h is the maximum packet

length and R_h is the link rate of the output link h, respectively. Thus, the service is guaranteed to complete no later than T_i + L_h/ R_h.

The approximation follows the architecture of C-SCORE. In other words, equations (3) and (4) are still used. However (5) is replaced by the following equation (7).

$$d_h(p) = (L_h^{max})/R_h + (n_{(f,h)}+1)S_h + delta_h(p), \quad (7)$$

7.4. E2E latency bound of the approximate C-SCORE

The E2E latency of a network with the approximate C-SCORE schedulers is upper bounded by

$$B/r + \sum_{h=0}^{H}\{(n_{(f,h)}+1)S_h + L_h/R_h\} + \sum_{h=0}^{H-1}\{delta_h(p)\},$$

where S_h is the slot length at node h, n_(f,h) is an integer specific to flow f at node h, which meets $(n_{(f,h)}-1)S_h < L_f/r \leq n_{(f,h)}S_h$. In other words, $n_{(f,h)}=ceiling[L_f/(r*S_h)]$. The term $\sum_{h=0}^{H-1}\{delta_h(p)\}$ includes the clock discrepancies and the propagation delays between the nodes. Considering that the clock discrepancies can be neglected in an absolute time, this term actually represents the E2E propagation delay.

8. Considerations for non-compliant nodes

There can be non-compliant end nodes and relay nodes in the network. There can be naturally end nodes without necessary signalling capabilities for admission control. There also can be legacy nodes or the nodes with dataplane enhancement solutions other than C-SCORE. How these can be compensated, or how much these nodes degrade the performance of C-SCORE will be discussed in a later version.

9. IANA Considerations

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

10. Security Considerations

This section will be described later.

11. Acknowledgements

12. Contributor

13. References

13.1. Normative References

[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-05, 8 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-dataplane-taxonomy-05>.

[I-D.ietf-mpls-mna-detnet]
Song, X., Mirsky, G., Varga, B., Gandhi, R., and Q. Xiong, "MPLS Network Action for Deterministic Networking", Work in Progress, Internet-Draft, draft-ietf-mpls-mna-detnet-00, 7 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-mpls-mna-detnet-00>.

[I-D.ietf-mpls-mna-ps-hdr]
Rajamanickam, J., Gandhi, R., Zigler, R., and J. Dong, "Post-Stack MPLS Network Action (MNA) Solution", Work in Progress, Internet-Draft, draft-ietf-mpls-mna-ps-hdr-06, 20 January 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-mpls-mna-ps-hdr-06>.

[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>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

13.2. 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.

[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.

[Y.3129] International Telecommunication Union, "Requirements and framework for stateless fair queuing in large scale networks including IMT-2020 and beyond", ITU-T Recommendation Y.3129, April 2024.

[Y.3148] International Telecommunication Union, "Functional architecture for stateless fair queuing in large scale networks including IMT-2020 and beyond", ITU-T Recommendation Y.3148, August 2025.

[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