DetNet Working Group
Internet-Draft
Intended status: Standards Track
Expires: 25 June 2026

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

22 December 2025

Latency Guarantee with Stateless Fair Queuing
draft-joung-detnet-stateless-fair-queuing-06

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 25 June 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. 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 6.3.7.

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 cases where separate information of L and r are not required, this alternative approach is more efficient. In Section 6.3.2, Section 6.3.4 and Section 6.3.5, we follow this approach.

6.3.2. Header format

6.3.2.1. IPv6 header format

In IPv6 header, a definition of an extension header called forwarding header will be specified in a later version of this draft. How a multiple of 8 byte can be used for two metadata, Fh(p) and L/r, will be specified as well.

6.3.2.2. MPLS label format

How 20 bits of the label value in a 4 byte label, and their stack are used for carrying two metadata, Fh(p) and L/r, will be specified in a later version.

6.3.3. 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.

6.3.4. 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

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.

The protocol for obtaining the departure time from node h-1, C(h-1)(p), at node h will be elaborated in a later version of this draft. The C(h-1)(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. Approximation of C-SCORE with 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 approximating C-SCORE with SP schedulers are specified. The E2E latency bound of the approximated system is also specified.

7.2. Switching node architecture

The architecture of a switching node, in which the SP scheduler with a limited number of queues behaves as an approximated priority queue, will be specified in a later version of the draft.

7.3. Algorithms

The algorithms for the delay factor calculation, the finish time update, and the queue management will be specified in a later version.

7.4. E2E latency bound of the approximated C-SCORE

The revised E2E latency bound of the approximated C-SCORE system, which can be very similar with that of the original C-SCORE, will be specified in a later version.

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-04, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-detnet-dataplane-taxonomy-04>.

[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>.

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.

[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