Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

A non-work conserving stateless core fair queuing scheduler for scalable deterministic networking

Jinoo Joung
Sangmyung University
Yeoncheol Ryoo
Electronics and Telecommunications Research Institute
Juhyeok Kwon
juhuk98@gmail.com

Sangmyung University
Jeong-dong Ryoo
Electronics and Telecommunications Research Institute
Taesik Cheung
Electronics and Telecommunications Research Institute

Research Article

Keywords: Deterministic Networking, End-to-end latency, Jitter, Finish time, Flow state

Posted Date: August 31st, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10739107/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

A non-work conserving stateless core fair queuing scheduler for scalable deterministic networking

Jinoo Joung[1], Yeoncheol Ryoo[2†], Juhyeok Kwon[3*†], Jeong-dong Ryoo[2,4†], Taesik Cheung[2†]

[1]Department of Human-Centered Artificial Intelligence, Sangmyung University, Seoul, 03016, Korea.
[2]Electronics and Telecommunications Research Institute (ETRI), Daejeon, 34129, Korea.
[3*]Department of Artificial Intelligence and Informatics, Sangmyung University, Seoul, 03016, Korea.
[4]Information and Communication Engineering Major, University of Science and Technology, Daejeon, 34113, Korea.

*Corresponding author(s). E-mail(s): juhuk98@gmail.com;
Contributing authors: jjoung@smu.ac.kr; dbduscjf@etri.re.kr; ryoo@etri.re.kr; cts@etri.re.kr;
†These authors contributed equally to this work.

Abstract

Deterministic networking (DetNet) is to guarantee both latency and jitter bounds to participating flows over end-to-end (E2E) paths with scalability. While the determinism is relatively easy to obtain in small networks with low propagation delays, the solutions for large scale networks are not yet well established. In the IETF DetNet Working Group (WG) and ITU-T SG13, a work conserving stateless core fair queuing (C-SCORE) technique is being standardized. C-SCORE transfers the flow state information obtained in the entrance node by recording it in packets as metadata. The core nodes can update it by modifying with the flow and node specific parameters, and use it for scheduling. The essence of the flow state information is the finish time (FT) of a packet, which is an ideal service completion time. The packets in the queue are served in the ascending order of the FT. It was proven that the E2E latency bound of C-SCORE is identical to that of the stateful fair queuing techniques such as the packetized generalized processor sharing (PGPS). However, because the packets are served whenever the link is idle, i.e. in work conserving manner, the jitter is not guaranteed in C-SCORE. We introduce a non-work conserving version of the stateless core fair queuing technique, called N-SCORE, which was adopted as a working group draft in IETF DetNet WG recently. We derive the analytical expressions for the E2E latency and jitter bounds, formally proving that N-SCORE preserves the optimal E2E latency of stateful fair queuing and C-SCORE while delivering superior jitter suppression. Furthermore, we characterize the nodal buffer requirements, establishing an upper bound of approximately 3NL, where N denotes the flow count and L the maximum packet size. These theoretical derivations are corroborated by extensive simulation results.

Keywords: Deterministic Networking, End-to-end latency, Jitter, Finish time, Flow state

1 Introduction

There are emerging applications that require both latency and latency variation (jitter) bounds in large-scale networks [1]. In environments such as smart factories and in-vehicle networks, there are demands for a network service that guarantees the upper bound of latency and jitter. Technologies to satisfy such quality of service (QoS) are being standardized in the IEEE 802.1 time-sensitive networking (TSN) task group [2], [3]. Additionally, to overcome the spatial limitations of TSN, the IETF Deterministic Networking (DetNet) working group (WG) [4] is working on the standardization of large-scale deterministic networking solutions.

In large-scale networks, end hosts may 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 [5], 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. 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 latencies [6].

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 [7]. 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 queue are served in an increasing order of the FT. Then the service of the packet is completed approximately the same as the FT. When this technique is applied, the worst latency at each node is proportional to the maximum packet length among flows sharing the link. 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 is a problem, so it is not usually adopted in practice. In [8], a framework for deriving FTs without maintaining flow states was presented, called C-SCORE. In this framework, the entrance node and core nodes are defined. 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.

In C-SCORE, the initial FT for a packet is obtained in the entrance node, and is updated in a core node by adding a nodal delay factor, which is a function of parameters that can depend on the node and the flow itself. The packets are to be put into a queue, if there is a packet currently being served. The packets in the queue are served in the increasing order of FTs. Therefore, in general the scheduler in C-SCORE requires a priority queue and is work conserving. It was proven that, for a certain choice of the delay factor function, the mathematical expression for the end-to-end (E2E) latency upper bound exists. It is remarkable that this E2E latency bound function is the same as the one with a network having stateful fair queuing schedulers in all the nodes [8]. It is well understood that the fair queuing schedulers are the best in terms of flow isolation and latency bound [7, 8]. Moreover, the bound is the function of the flow intrinsic parameters, except the ratio of the maximum packet length on a link to the capacity of the link. Therefore, the bound is adjustable with the flow's own parameters. A network can meet the requested E2E latency bound of a flow by allocating an appropriate service rate to the flow [7, 8].

While work-conserving mechanisms like C-SCORE ensure link efficiency, they inherently fail to provide deterministic jitter guarantees. To address this, we introduce N-SCORE, a non-work-conserving evolution of stateless core fair queuing, which has recently been adopted as a working group draft in IETF DetNet WG. N-SCORE achieves deterministic performance by incorporating an 'eligible time' parameter, calculated at the entrance and updated at core nodes to enforce a strict earliest-service-start-time constraint. We analytically demonstrate that N-SCORE maintains the E2E latency bound identical to stateful fair queuing and C-SCORE, while significantly improving jitter performance, a claim further validated through extensive simulations.

Furthermore, we characterize the nodal buffer requirements, establishing an upper bound of approximately 3NL, where N denotes the flow count and L the maximum packet size. This modest requirement significantly alleviates the implementation complexity typically associated with PIFO (Push-In First-Out) queuing.

The remainder of this paper is structured as follows: Section 2 reviews existing literature on fair queuing and IEEE 802.1 TSN scheduling mechanisms. Section 3 introduces the N-SCORE algorithm, provides a rigorous mathematical analysis, and derives its bounds for E2E latency, jitter, and

per-hop buffer occupancy. In Section 4, the performance of N-SCORE is validated through extensive simulations across diverse scenarios. Finally, Section 5 concludes the paper with a summary of our findings and future directions.

2 Related works

2.1 C-SCORE

The concept of treating flows as fluids and providing each flow with precisely the demanded service through appropriate scheduling at intermediate nodes has been proposed since the 1990s. Generalized processor sharing (GPS) proposed a paradigm that treats transmitted data as fluids and determines the ideal service order [7]. PGPS and WFQ, which implement this paradigm in a packet environment, have played a pioneering role in such packet-based schedulers [9–11]. The mathematical symbols used in this paper are presented in Table 1.

PGPS determines the service order of packets in ascending order of the FT, which is derived from the following equation:

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

Here, p represents the p^{th} packet of the flow, $A(p)$ denotes the arrival time of packet p at the node, $L(p)$ represents the length of p, and r signifies the service rate allocated to the flow to which

Table 1 Definitions of symbols used

Symbol	Quantity
p	p^{th} packet of the flow under observation.
h	h^{th} node in the path of the flow under observation
$F_h(p)$	finish time of packet p at node h
$A_h(p)$	arrival time of packet p at node h
$C_h(p)$	service completion time of packet p at node h
$C^*(p)$	service completion time of packet p at node h in an ideal preemptive node
$E_h(p)$	eligible time of packet p at node h
$L(p)$	length of packet p
L	max packet length of flow under observation
L^+	max packet length over all the high priority flows
$r, r(p)$	service rate of the flow under observation
L_h^{max}	max packet length over all the flows at node h
R_h	link capacity of node h
$d_h(p)$	delay factor function of packet p at node h
$\delta_h(p)$	time difference between node h and node $h+1$. It is defined as $A_{h+1}(p) - C_h(p)$

p belongs. $V(A(p))$ is referred to as the virtual time function, which is the amount of service that would have been received by a continuously backlogged flow with the allocated the service rate of 1 bps, starting from time 0. If the link is fully utilized, the flow with the service rate of 1 bps would actually receive 1 bps service. If the link is under-utilized, then the flow would receive much more than 1 bps, and the virtual time function would have a larger value. The virtual time function increases slowly in a fully utilized link and rapidly in an under-utilized link. Thus, the virtual time function reflects the system's service progress. The finish time $F(p)$ represents the fair finish time of p, assuming an ideal fluid environment. Nodes serve packets in the order of increasing $F(p)$ values. The key point of (1) is to ensure that in the worst-case scenario, where all flows are active and the

link is fully utilized, packets belonging to a flow are serviced at intervals of $L(p)/r$ compared to their preceding packets. Meanwhile, by employing a work-conserving scheduler, it prevents unused and wasted link resources.

In order to calculate (1), it is necessary to remember the FT of the previous packet, $F(p-1)$, of the flow. Upon packet arrival, it is essential to identify which flow it belongs to and determine the finish time of the most recent packet in that flow. The FT of this most recent packet represents what is commonly referred to as the "flow state." Remembering such state information entails significant complexity from the perspective of a core node managing millions of flows, and this has become the primary reason why schedulers in this PGPS series are not actually utilized on the Internet.

The computation of $V(t)$ also entails the complexity of tracking the currently serviced flows in real-time to calculate the sum of their r values. Considering that the initiation of flows can occur at a multitude of times within a short duration, this calculation is challenging. Therefore, instead of precisely computing $V(t)$, various approaches have been proposed to estimate it using simpler methods [10, 11]. Among these, the virtual clock (VC) algorithm [10] determines finish times using the current time t rather than $V(t)$.

The recently proposed stateless fair queuing technique, known as C-SCORE [8], which is approved as ITU-T Y.3129 standard [12] and actively being discussed in the IETF DetNet WG [13] as of Sep. 2024, utilizes the virtual clock algorithm. This method determines the finish time at each node using packet metadata, or equivalently packet header fields, without requiring state management at the core node. Consequently, it decides the service order of packets based on these finish times.

In C-SCORE, the FT of packet p at node h, denoted as $F_h(p)$, is determined using the following method. Initially, if we designate the ingress node as 0, then the finish time $F_0(p)$ at node 0 is calculated as follows.

$$F_0(p) = \max\{F_0(p-1),\ A_0(p)\} + L(p)/r. \quad (2)$$

$F_0(1)$, the finish time of the first packet of a flow at node 0, is defined to be 0. The finish time $F_h(p)$ at node h after the initial node is calculated as follows.

$$F_h(p) = F_{h-1}(p) + d_{h-1}(p). \quad (3)$$

Here, $d_h(p)$ is referred to as the delay factor, which can be presented through various functions. For instance, the delay factor can be computed using

(4).

$$d_h(p) = \frac{L_h^{max}}{R_h} + L/r. \quad (4)$$

To calculate $F_h(p)$ using (3), it is necessary to have $F_{h-1}(p)$ and $d_{h-1}(p)$. These values or the values used to calculate them must be recorded as packet p's metadata at node h-1 to be used at the next node, h. For instance, when calculating the delay factor as shown in (4), the metadata values of L and r must be recorded. L is the maximum packet length of the flow to which the packet belongs. It should be noted that L_h^{max} and R_h are values maintained by the node h, not a part of the metadata.

C-SCORE [8, 12, 13] has garnered attention for its ability to provide the considerable flow isolation and the ideal E2E latency bound, without maintaining the flow states. Alternatively, C-SCORE necessitates a priority queue, an approach that offers lower implementation complexity compared to flow-state tracking. While inserting or updating flow states can expedite finish time calculations with $O(1)$ computational complexity via hash structures, the overhead of managing flow states requires tracking active flows in real time, yielding a worst-case complexity of $O(N)$. Given that the flow count N at a core node can grow boundlessly, a $O(N)$ overhead renders practical implementation unfeasible.

Conversely, a priority queue entails a complexity of $O(\log M)$, where M denotes the number of packets buffered in the queue. Because M is constrained by the physical buffer capacity, the maximum execution time remains deterministic under worst-case conditions. Although the achievable line rate may vary depending on the implementation specification of the priority queue, addressing this issue lies beyond the scope of this work.

2.2 Core Jitter Virtual Clock (CJVC)

The FQ scheduling technique stores state information for each flow, extracts this information when a packet of the corresponding flow arrives, calculates the FT of the packet, and stores it again. If $F(p-1)$ in (1) represents state information, the newly calculated $F(p)$ becomes the updated state of the flow. This complexity is the reason why it is difficult to implement FT-based scheduling in a core node where millions of flows are active simultaneously. Therefore, if $F_h(p)$ is the FT of p at the h^{th} node in the path, methods of estimating $F_h(p)$ with $F_0(p)$ instead of $F_h(p-1)$ have been proposed. These techniques are collectively called core-stateless fair queuing (CSFQ) [14, 15]. In particular, in the study of Stoica et. al. [14] , the seminal work of similar studies, $F_h(p)$ is derived in the following way. First, $E_h(p)$ is defined as the eligible time of the packet p to start the service. Each packet becomes a candidate for service after

the eligible time has elapsed. The packet having the lowest $F_h(p)$ value among these candidates is served.

$$
\begin{aligned}
E_0(p) &= A_0(p), \\
E_h(p) &= A_h(p) + G_{h-1}(p) + \varepsilon_h(p), \\
F_h(p) &= E_h(p) + \frac{L(p)}{r}.
\end{aligned} \tag{5}
$$

Here, $G_h(p)$ is the latency margin at h (the difference between the FT and the actual service completion time), and $\varepsilon_h(p)$ is the latency that forces $E_h(p)$ to be always greater than $E_{h-1}(p)$. This term guarantees the service order preservation between packets in the flow. Consequently, $E_h(p)$ can be expressed as a function of $E_0(p)$. That is, if $G_h(p)$, r, $\varepsilon_h(p)$, and $L(p)$ are stored in the header of the packet and sent to the next node, the eligible time and FT of (4) can be derived without managing the flow state information. This scheduling method, CJVC, is based on the VC [10], and is claimed to reduce jitter as well. However, compared to C-SCORE and N-SCORE, CJVC requires more metadata to carry, thus has a larger complexity in calculation of FT. Moreover, the service completion time of a packet has to be known before its actual service completion, in order to calculate $G_h(p)$ and put it as metadata, which is too burdensome for a real-time per-packet processing. This difficulty can be illustrated by the example case of Precision Time Protocol (PTP). PTP uses a second packet, called a Follow_Up message, in two-step mode because it is practically impossible for standard hardware and software to know the exact departure timestamp before the packet leaves the physical network interface.

2.3 IEEE 802.1 TSN Standards

In this subsection, we discuss the IEEE 802.1 TSN standard technologies, focusing on jitter guarantee through Time-Aware Shaper (TAS) and Cyclic Queuing and Forwarding (CQF).

IEEE 802.1Qbv, TAS, is a mechanism that introduces Time Division Multiple Access (TDMA) concepts into Ethernet switching to ensure deterministic delivery for high-priority traffic [16]. The key mechanism for TAS is the Gate Control List (GCL), which operates via a "gate" mechanism in front of each egress queue. Gates open and close according to a precise schedule, allowing only specific traffic classes to transmit during designated time slots. By utilizing guard bands or strict scheduling, it physically prevents Best-Effort (BE) traffic from interfering with Scheduled Traffic (ST), eliminating the head-of-line blocking. Since transmission timing is dictated by a nanosecond-precision schedule, queuing jitter within the switch is theoretically reduced to near-zero levels. However, determining an optimal GCL for an entire network is an NP-hard problem [17]. As the number of nodes and streams increases,

the computational overhead for the Centralized Network Configurator (CNC) grows exponentially. Any change in network topology or the addition of a new flow requires a complete recalculation and redistribution of GCLs across all affected switches. Moreover, the use of guard bands to prevent frame truncation can lead to idle times on the link, reducing the effective throughput for non-scheduled traffic.

IEEE 802.1Qch, Cyclic Queuing and Forwarding (CQF), simplifies jitter control by synchronizing transmissions into discrete cycles rather than individual frame schedules [18]. The key mechanism for CQF is the so-called double buffering mechanism, in which nodes utilize a ping-pong buffering strategy with even/odd cycles. Frames received during cycle n are buffered and strictly transmitted during cycle (n+1). The E2E latency is bounded by the number of hops (H) and the cycle time (T). The total latency is consistently within the range of (H-1)T to (H+1)T [19]. Unlike TAS, CQF does not require micro-management of individual frame offsets, making it significantly easier to deploy in large-scale environments. However, while TAS offers nanosecond-level precision, CQF jitter is bounded by the cycle time (T). A frame can be sent at any point within its designated cycle, resulting in higher accumulated jitter compared to TAS. CQF also requires very strict phase synchronization between adjacent nodes. Any drift beyond the cycle boundary can result in frames missing their output window, leading to packet loss or timing violations. Therefore, there is a fundamental trade-off, with which shorter cycles reduce jitter but limit the maximum frame size (MTU) that can be transmitted, while longer cycles increase the overall latency.

In the recent research, the focus is shifting toward IEEE 802.1Qcr, Asynchronous Traffic Shaping (ATS) [20]. ATS aims to provide per-hop jitter bounds without requiring global time synchronization, addressing the scalability limitations found in both TAS and CQF.

It uses an Urgency-Based Scheduler (UBS) or Interleaved Regulator (IR) that calculates an eligibility time for each frame based on its arrival and token bucket state. By reshaping traffic at every hop, ATS prevents the burst accumulation that typically occurs in standard switched Ethernet, effectively bounding the jitter without a global clock. Thus. It is highly scalable; if a link fails or a node is added, it does not require a network-wide recalculation of a schedule [21]. ATS is the best for statistical guarantees in massive networks where maintaining nanosecond synchronization across thousands of nodes is technically or economically unfeasible. However, because the UBS, or the IR has to put the packets from different flows into a single FIFO queue for reshaping, the per-hop latency bound of a flow increases to

the level of the worst latency bound of a flow within the FIFO [22].

3 Proposed scheduler

3.1 Algorithm

For readability, the essence of the fair queuing scheduler and C-SCORE is repeated with the equations below.

$$\text{FQ} : F(p) = max\{F(p-1),\ V(A(p))\} + L(p)/r. \tag{1}$$

$$\text{C-SCORE} : F_h(p) = F_{h-1}(p) + d_{h-1}(p). \tag{3}$$

The key idea in (3) was to use $F_0(p)$ instead of $F_h(p-1)$ in (1) to obtain $F_h(p)$. For example, suppose a flow continues to send packets and is continuously backlogged on the entrance node. In this case, $F_0(p)$ is calculated as $F_0(p-1)+L(p)/r$. That is, the interval between the consecutive packets' FTs is maintained at $L(p)/r$. These characteristics are independent of the states of other flows. That is, the FT of a packet can be inferred, ahead of traveling to the downstream nodes, only from the packet lengths and the assigned service rate of the flow as $L(p)/r$ during the backlogged period. Even in the downstream nodes, during the backlogged periods, we can predict $F_h(p) = F_{h-1}(p)+d_{h-1}(p)$, where $d_h(p)$ is some delay function specific to the node and the packet. Note that $d_h(p)$ may not be a function of p, but of p's flow. In this case, it can maintain the intervals between packets' FTs identical to those of $F_0(p)$.

C-SCORE behaves in a work conserving manner. The link never idles when there are packets. The lower bound in E2E latency is only dictated by the propagation and transmission delays. Thus the latency variation, the jitter, can be almost as large as the E2E latency upper bound. We introduce a non-work conserving version of the C-SCORE, called N-SCORE.

In N-SCORE, an additional parameter called the eligible time of a packet is introduced. The eligible time of a packet is obtained at the entrance node, recorded in the packet, modified in core nodes, and used as the earliest packet service start time allowed. Since packets arrived earlier than its eligible time cannot be served even when the link idles, N-SCORE is non- work conserving.

In particular, the eligible time, $E_h(p)$, and the finish time, $F_h(p)$, of packet p at node h are obtained as in the following.

$$E_0(p) = \max\{F_0(p-1),\ A_0(p)\}. \tag{6}$$

$$E_h(p) = E_{h-1}(p) + d_{h-1}(p),\ for\ h > 0, \tag{7}$$

where $d_h(p) = \frac{L_h^{max}}{R_h} + L/r + \delta_h(p)$. $\delta_{h-1}(p)$ is called the time difference between the nodes $(h-1)$

and h, including the clock difference and the propagation delay. $\delta_{h-1}(p) = A_h(p) - C_{h-1}(p)$, where $C_{h-1}(p)$ is the service completion time of p at node $h-1$.

$$F_h(p) = E_h(p) + \frac{L(p)}{r}, \ for\ h \geq 0. \tag{8}$$

A packet can join the output link scheduler right after its eligible time. If there is no packet 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 FTs.

Note that the delay factor in (4) is also used in (7). The FT of N-SCORE is identical to that of C-SCORE, since from (7) and (8), (3) can be derived. In other words, from (8),

$$\begin{aligned} F_h(p) &= E_h(p) + \frac{L(p)}{r} \\ &= E_{h-1}(p) + d_{h-1}(p) \ + \frac{L(p)}{r} \\ &= F_{h-1}(p) + d_{h-1}(p), \end{aligned} \tag{9}$$

which concludes the same as (3). Therefore, we can say that the packets in N-SCORE have the same service order as those in C-SCORE. The only difference in two systems is the existence of the eligible times.

3.2 Latency and jitter bounds

In this subsection, the E2E latency and jitter bounds of a network with the proposed schedulers will be derived. We start by the definition of the service interval, which is a virtual interval during which a packet is to be serviced.

Definition 1 The service interval of packet p at node h is defined to be $(E_h(p), F_h(p)]$.

Figure 1 depicts the arrival and the service intervals of packets from a flow. The thick red arrows indicate the service intervals of packets. Figure 2 depicts the service intervals in adjacent nodes, in which the service intervals shift with a fixed amount of time.

The following lemma about the service intervals holds in N-SCORE.

Lemma 1 The service intervals of packets from a flow do not overlap with each other.

Proof From (6), $F_0(p-1) \leq E_0(p)$, thus the service intervals of consecutive packets at node 0 do not overlap. Moreover, from (7) and (9),

$$\begin{aligned} F_h(p-1) &= F_0(p-1) + \sum_{i=0}^{h-1} d_i(p) \\ &\leq E_0(p) + \sum_{i=0}^{h-1} d_i(p) = E_h(p). \end{aligned}$$

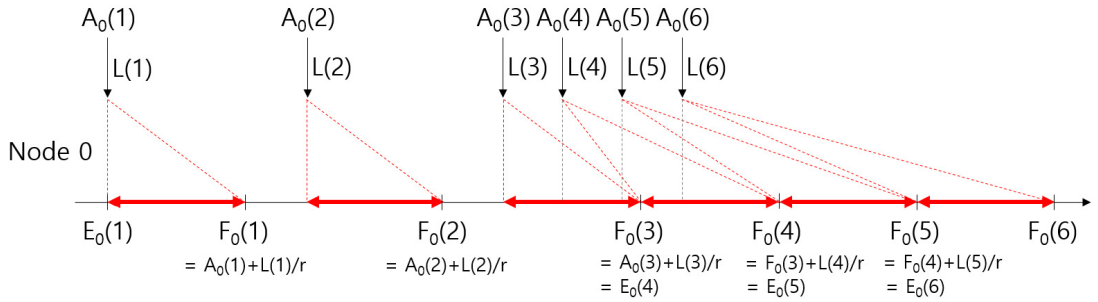


Fig. 1 Illustration of the arrival times, $A_0(p)$, with packet lengths, $L(p)$, and the service intervals (red bold arrows) of packets at the entrance node 0

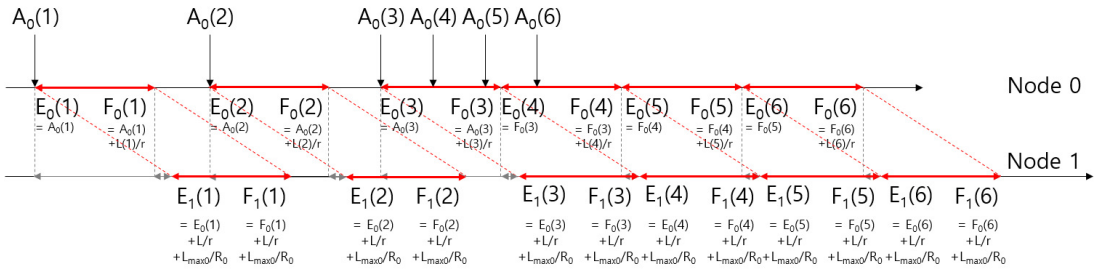


Fig. 2 Illustration of the service intervals at the entrance node and the next node

Thus, $(E_h(p-1), F_h(p-1)]$ does not overlap with $(E_h(p), F_h(p)]$, since $F_h(p-1) \leq E_h(p)$. This inequality holds for any consecutive packets at any node. The lemma follows. □

From now on we will show that N-SCORE guarantees the same E2E latency bound with C-SCORE. First, we will introduce the concept of the ideal preemptive system. A perfectly preemptive system preempts a packet without wasting the already served portion.

Consider a perfectly preemptive system, in which packet p's actual service completion time is denoted as $C^*(p)$. Its $F(p)$, $E(p)$, $A(p)$ are the same as those of the real non-preemptive system. Only the actual service completion time $C^*(p)$ can be different from that of the real non-preemptive system, $C(p)$. We will focus on a single core node, and omit the subscript h that denotes the node, for simplicity.

From Theorem 1 to Lemma 5, we will assume that the arrival time of any packet to a node is earlier than the packet's eligible time, that is $A(p) \leq E(p)$. We will prove that this is indeed true in Lemma 6.

Theorem 1 *If $A(p) \leq E(p)$ for any packet p, then the actual service completion time of p is less than or equal to the FT of p, that is $C^*(p) \leq F(p)$. In order to prove Theorem 1, we need the following lemmas. Let $(k-1), k, (k+1)...$ be the packets within a node, regardless of the flows they belong to, which are sorted according to their FTs. $r(k)$ is the service rate of the flow, to which the packet k belongs.*

Lemma 2 Assume the condition $A(p) \leq E(p)$ is met for any p. At the FT of the previous packet $k-1$, $F(k-1)$, a portion of the packet under observation, k, is already served at least as much as $r\{F(k-1) - E(k)\}^+$, where $y = \{x\}^+$ denotes the function $y = x \; if \; x \geq 0, \; 0 \; otherwise.$

Proof If $F(k-1) \leq E(k)$, then $\{F(k-1) - E(k)\}^+ = 0$ and it is trivial to show. Otherwise, i.e. if $F(k-1) > E(k)$, assume at time 0 the system initiates. Let t_0, $t_0 \in [0, \; F(k-1)]$, be the last moment closest to $F(k-1)$, at which the system idles. Then the work done by the system from t_0 to t, defined as $W(t_0, \; t)$, at any time $t > t_0$, is equal to $R(t - t_0)$, where R is the link capacity of the system. It is because during $[t_0, t]$ the system never idle.

Further let G be the set of packets whose services are completed during $(t_0, F(k-1)]$. Also let Gi be the subset of G, whose packets are from the flow i. $G1, ..., Gn$ partition G, where there are n flows. We claim that

$$\sum_{j \in G} L(j) + r(k)\{(F(k-1) - E(k)\} \leq \{F(k-1) - t_0\}R, \tag{10}$$

where $r(k)$ is the service rate of the flow to which packet k belongs.

Note that, from (8), $L(j)/r(j) = F(j) - E(j)$ for any packet j. Also note that for packets within the same flow the service intervals do not overlap, from Lemma 1. Let $r_1, r_2, ..., r_n$ be the service rates of the flows. Then,

$$\begin{aligned}
\sum_{j \in G} L(j) &= \sum_i \sum_{j \in Gi} L(j) \\
&= \sum_i \sum_{j \in Gi} (F(j) - E(j))r_i \\
&= \sum_i (\sum_{j \in Gi} \{F(j) - E(j)\})r_i. \qquad (11)
\end{aligned}$$

Let $r(k) = r_k$, $1 \leq k \leq n$, without loss of generality. Then, from (11),

$$\begin{aligned}
&\sum_{j \in G} L(j) + r_k\{(F(k-1) - E(k)\} \\
&= \sum_i (\sum_{j \in Gi} \{F(j) - E(j)\})r_i + r_k\{F(k-1) - E(k)\} \\
&= \sum_{i, i \neq k} (\sum_{j \in Gi} \{F(j) - E(j)\})r_i \\
&\quad + (F(k-1) - E(k) + \sum_{j \in Gk} \{F(j) - E(j)\})r_k \\
&\leq \sum_i (F(k-1) - t_0)r_i \\
&\leq (F(k-1) - t_0)R,
\end{aligned}$$

since $\sum_i r_i \leq R$, because of the admission condition. Note that the sum of service intervals of packets, whose FT is less than or equal to $F(k-1)$, from a single flow does not exceed the interval $(t_0, F(k-1)]$, because they do not overlap, by Lemma 1. Therefore, the sum of the lengths of packets, whose FT is less than or equal to $F(k-1)$, plus $r_k\{F(k-1) - E(k)\}$ does not exceed $(F(k-1) - t_0)R$. This inequality is illustrated in Figure 3. (10) follows.

After the service completion of the packet $(k-1)$, the packet k has the highest priority to be served within the system. There are two cases.

Case 1: There is no packet served, even partially, before k and after $(k-1)$. It means that any other

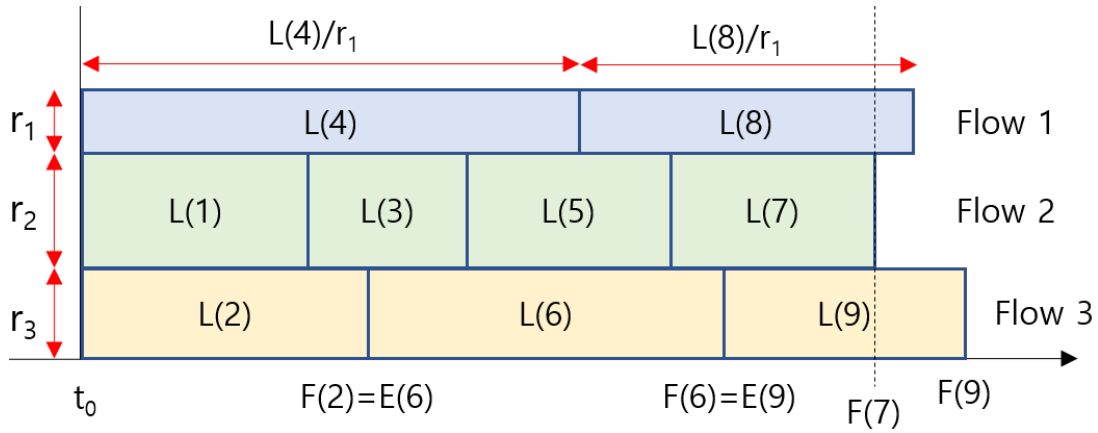


Fig. 3 Illustration of the inequality between the sum of $L(j)$s plus $\{F(7) - E(8)\}r_1$ and the $\{F(7) - t_0\}R$. $L(j)$ is the length of packet j and indicated by the volume of a rectangle. In this worst example, the flows generate packets as much as they can. The sum of packet lengths $L(1) + \cdots + L(7)$, whose FTs are less than or equal to $F(7)$, plus $\{F(7) - E(8)\}r_1$ cannot exceed the $\{F(7) - t_0\}R$ because $r_1 + r_2 + r_3$ is less than or equal to the link capacity, R.

packet in G must have been served before the service start of packet k. Thus, the work that has been done to packet k during $(t_0,\ F(k-1)]$, $W_k(t_0,\ F(k-1)) = W(t_0,\ F(k-1)) - W_G(t_0,\ F(k-1)) = F(k-1) - t_0R - \sum_{j\in G} L(j) \geq r(k)\{F(k-1) - E(k)\}$, from (10).

Case 2: There are packets served partially before k and after $(k-1)$. The work that has been done to packet k during $(t_0,\ F(k-1)]$, $W_k(t_0,\ F(k-1))$, is equal to $R\{F(k-1) - E(k)\}$. In other words, after $E(k)$, the packet k is guaranteed to be served with the link capacity, since at $E(k)$ all the packets in the queue have lower priority than the packet k.

In both cases, $W_k(t_0,\ F(k-1)) \geq r(k)\{F(k-1) - E(k)\}$. This proves the lemma. □

Lemma 3 Assume the condition $A(p) \leq E(p)$ is met for any p. The service of packet p in the preemptive system is completed not later than $E_h(p) + \frac{L(p)}{r(p)}$. In other words, $C^*(p) - E(p) \leq L(p)/r(p)$.

Proof From Lemma 2, a portion of at least $r\{F(k-1) - E(k)\}^+$ of packet k is already served at $F(k-1)$.

Case 1: $F(k-1) > E(k)$. After the FT of the previous packet $k-1$, $F(k-1)$, the packet under observation, k, is served at the rate of R, the link capacity, thus its remaining portion $L(k) - r(k)\{F(k) - E(k)\}$ is served during $[L(k) - r(k)\{F(k) - E(k)\}]/R$. Thus, the total time to serve k is $F(k) - E(k) + [L(k) - r(k)\{F(k) - E(k)\}]/R \leq F(k-1) - E(k) + [L(k) - r(k)\{F(k-1) - E(k)\}]/r(k) = L(k)/r(k)$. The service completion time does not exceed $E(k) + L(k)/r(k)$.

Case 2: $F(k-1) \leq E(k)$. In this case at $E(k)$, the packet k has the highest priority in the queue, therefore the packet k is served with the rate R. $E(k) + L(k)/R$ is its service completion time.

In both cases, the lemma holds. □

Proof of Theorem 1 Theorem 1 follows directly from Lemma 3, because from (8), $F(p) = E(p) + L(p)/r(p)$ for any p, where $r(p)$ is the service rate of the flow, to which the packet p belongs. Thus, $C^*(p) \leq F(p)$ for any p. □

Now consider a real non-preemptive system. In the real system, the service order can be different from the FT order. Let's denote packet p's service completion time as $C(p)$. The following theorem states the nodal latency bound of N-SCORE in a real non-preemptive system.

Definition 2 System busy interval (SBI) is defined to be a maximal time interval during which the server (scheduler) never idle.

Lemma 4 The start time and end time of an SBI, and the packets served within an SBI are identical for a non-preemptive system (NPS) and the corresponding ideal preemptive system (PS), for any SBI.

Proof The arrival processes are identical for both systems. A system becomes busy when the first packet arrives to an idle system. Let this time be $t0$. The system becomes idle when the sum of the arrived traffic over all the flows, $\sum_i A_i(t_0,\ t)$, equals to the amount of service given within $[t0, t]$, $W(t_0,\ t)$. That is, the end time of the SBI, t_E, is the minimum t that makes $\sum_i A_i(t_0,\ t) = W(t_0,\ t)$. $W(t_0,\ t) = (t - t_0)R$, for t within the SBI, where R is the link capacity. Since both $A_i(t_0,\ t)$ and $W(t_0,\ t)$ are identical for NPS and PS, the $t_E$is identical for both systems. The lemma follows. □

Theorem 2 *If $A(p) \leq E(p)$ for any packet p, then the service completion time of any packet p in the real system $C(p) \leq C^*(p) + L^{max}/R$, where L^{max} is the maximum packet length within the system.*

Proof The proof is similar to that of Theorem 1 of [23]. Consider a SBI. Because of Lemmas 2 and 3, it is sufficient to prove for all the packets in a SBI. Let us say t_0 is the start time of the SBI. Consider the packet of interest, p.

Case 1: There are packet(s) j such that $C^*(p) < C^*(j)$ are served in the interval $[t_0,\ C(p)]$, in NPS.

Consider the service start times of packets j in NPS such that$C^*(p) < C^*(j)$. Let's say t_1 is the largest such time less than $C(p)$. Let j be the packet that starts service at t_1. Also let t_2 be the least time greater than t_1 such that packet(s) k gets services in $[t_2,\ C(p)]$, where $C^*(k) < C^*(p)$. By definition of t_1 and t_2, they can be separated at most by a maximum packet service time, that is $t_2 - t_1 \leq \frac{L^{max}}{R}$. Remember that we are considering a single SBI. Let K be the set of packets which are served in $[t_2,\ C(p)]$. Note that K includes p. Because at t_1 the lower priority packet started the service, all the packets in K must have arrived after t_1. In other words, $\min_k\{A(k)\} > t_1$.

Note that $C^*(p)$ is defined as the service completion time of p in PS. By definition of K, all the other packets in K are serviced before $C^*(p)$ in PS. Therefore, $C^*(p) \geq t_1 + \sum_{k\in K} \frac{L(k)}{R}$.

Since $t_2 - t_1 \leq \frac{L^{max}}{R}$, $C^*(p) \geq t_2 - \frac{L^{max}}{R} + \sum_{k\in K} \frac{L(k)}{R}$. Because the service order in NPS among packets K can be different from the service order in PS, $C(p) \leq t_2 + \sum_{k\in K} \frac{L(k)}{R}$, and $C^*(p) \geq C(p) - \frac{L^{max}}{R}$.

Case 2: No packet j such that $C^*(p) < C^*(j)$ are served in the interval $[t_0,\ C(p)]$, in NPS.

Let K be the set of packets served in $[t_0,\ C(p)]$. PS guarantees that p is transmitted by $C^*(p)$, where $C^*(p) = t_0 + \sum_{k \in K} \frac{L(k)}{R}$.

Moreover, in this case $C^*(p) \geq C(p)$, because the service order in NPS can be different from that in PS.

The theorem follows in both cases. □

Theorem 2 formalizes the property that a packet incurs non-preemptive interference from lower-priority traffic at most once.

Lemma 5 If $A(p) \leq E(p)$ for any packet p, then $C(p) \leq F(p) + \frac{L^{max}}{R} = E(p) + \frac{L(p)}{r} + \frac{L^{max}}{R}$.

Proof From Theorem 1 and Theorem 2, Lemma 5 is directly obtained. □

Now consider the condition $A(p) \leq E(p)$ that was required in the lemmas and the theorems so far, except Lemma 1. In the following lemma, we prove that the condition is actually met for any packet at any node. That is, the arrival of a packet happens before its eligible time at a node.

Lemma 6 $A_h(p) \leq E_h(p)$ for any p and h.

Proof The proof is by induction. We first show it when $h = 0$. From (6), $E_0(p) = \max\{F_0(p-1),\ A_0(p)\} \geq A_0(p)$. Now assume that the inequality is true at node $(h-1)$. Because of this assumption, both Theorem 2 and Lemma 5 hold at node $(h-1)$. From (7), $E_h(p) = E_{h-1}(p) + \frac{L_{h-1}^{max}}{R_{h-1}} + \frac{L}{r} + \delta_{h-1}(p)$, for $h > 0$. From Lemma (5), $E_h(p) = E_{h-1}(p) + \frac{L_{h-1}^{max}}{R_{h-1}} + \frac{L}{r} + \delta_{h-1}(p) \geq C_{h-1}(p) + \delta_{h-1}(p) = A_h(p)$, for $h > 0$. Note that $C_{h-1}(p) + \delta_{h-1}(p) = A_h(p)$ by definition of $\delta_{h-1}(p)$. With the above inequality, we just showed that $A_h(p) \leq E_h(p)$ given $A_{h-1}(p) \leq E_{h-1}(p)$. The lemma follows. □

From Lemma 6, the condition $A(p) \leq E(p)$ holds true, thus Theorem 1, Theorem 2, and Lemma 5 can be applied to any packet at any node in a network with N-SCORE.

Now we are ready to state the main contributions of this paper. Let H be the last node of the flow under observation.

Theorem 3 *The E2E latency of N-SCORE is upper bounded by* $\frac{B-L}{r} + \sum_{h=0}^{H}\{\frac{L}{r} + \frac{L_h^{max}}{R_h}\} + \sum_{h=0}^{H-1} \delta_h(p)$, *where B, L, and r are the maximum burst size, the maximum packet length, and the service rate of the flow under observation.*

Proof By Lemma 3, $C^*(p) - E(p) \leq L(p)/r(p)$. By Theorem 2, $C(p) \leq C^*(p) + L^{max}/R$. Therefore, $C(p) \leq E(p) + \frac{L(p)}{r(p)} + \frac{L^{max}}{R}$. Thus, $C_h(p) \leq E_h(p) + \frac{L(p)}{r(p)} + L_h^{max}/R_h$, for h, $0 \leq h \leq H$.

On the other hand, from (7), $E_h(p) = E_{h-1}(p) + d_{h-1}(p)$ for h, $h > 0$. Thus,

$$\begin{aligned} C_H(p) &\leq E_H(p) + \frac{L(p)}{r(p)} + \frac{L_H^{max}}{R_H} \\ &= E_{H-1}(p) + d_{H-1}(p) + \frac{L(p)}{r(p)} + \frac{L_H^{max}}{R_H} = \dots \\ &= E_0(p) + d_0(p) + d_1(p) + \dots \\ &\quad + d_{H-1}(p) + \frac{L(p)}{r(p)} + \frac{L_H^{max}}{R_H} \end{aligned}$$

$$= E_0(p) + \sum_{h=0}^{H-1}\{\frac{L}{r(p)} + \frac{L_h^{max}}{R_h} + \delta_h(p)\} + \frac{L(p)}{r(p)} + \frac{L_H^{max}}{R_H}. \quad (12)$$

Note that L is the maximum packet length within the flow, while $L(p)$ is the length of the packet p. $E_0(p) = \max\{F_0(p-1),\ A_0(p)\}$, from (6). Since p can be the last packet of a burst with size $B(p)$, which arrived at once to the network in the worst case,

$$E_0(p) - A_0(p) \le \frac{B(p) - L(p)}{r(p)}. \quad (13)$$

Therefore, from (12) and (13), the E2E latency of p, $D(p) = C_H(p) - A_0(p) \le \frac{B(p)}{r(p)} + \sum_{h=0}^{H-1}\{\frac{L}{r(p)} + \frac{L_h^{max}}{R_h} + \delta_h(p)\} + \frac{L_H^{max}}{R_H} = \frac{B(p)-L}{r(p)} + \sum_{h=0}^{H}\{\frac{L}{r(p)} + \frac{L_h^{max}}{R_h}\} + \sum_{h=0}^{H-1}\delta_h(p)$. The theorem follows. □

Note that the E2E latency upper bound of N-SCORE is identical to that of C-SCORE. Also note that the term $\sum_{h=0}^{H-1}\delta_h(p)$ includes the clock differences and the propagation delays between the nodes. Considering that the clock difference can be neglected in an absolute time, this term actually represents the E2E propagation delay.

Theorem 4 *The E2E latency of N-SCORE is lower bounded by* $\sum_{h=0}^{H-1}\{\frac{L}{r} + \frac{L_h^{max}}{R_h} + \delta_h(p)\} + \frac{L^{min}}{R_H}$, *where* L, L^{min}, *and* r *are the maximum packet length, minimum packet length, and the service rate of the flow under observation.*

Proof By definition of the eligible time, a packet cannot start its service before $E_H(p)$ at the last node. Thus,

$$\begin{aligned} C_H(p) &\ge E_H(p) + \frac{L^{min}}{R_H} \\ &= E_{H-1}(p) + d_{H-1}(p) + \frac{L^{min}}{R_H} = \dots \\ &= E_0(p) + d_0(p) + d_1(p) + \dots + d_{H-1}(p) + \frac{L^{min}}{R_H} \\ &= E_0(p) + \sum_{h=0}^{H-1}\{\frac{L}{r(p)} + \frac{L_h^{max}}{R_h} + \delta_h(p)\} + \frac{L^{min}}{R_H}. \end{aligned}$$

Note that $E_0(p) \ge A_0(p)$. As such,

$$\begin{aligned} D(p) &= C_H(p) - A_0(p) \\ &\ge \sum_{h=0}^{H-1}\{\frac{L}{r(p)} + \frac{L_h^{max}}{R_h} + \delta_h(p)\} + \frac{L^{min}}{R_H}. \end{aligned}$$

The theorem follows. □

The following lemma states the jitter bound of N-SCORE.

Lemma 7 The E2E jitter of N-SCORE is bounded by $\frac{B}{r} + \frac{L_H^{max}}{R_H} - \frac{L^{min}}{R_H}$.

Proof We define the jitter of a flow as the difference between the maximum and minimum latencies the flow experiences. The jitter is thus bounded by the difference between the upper bound and the lower bound of E2E latency. From Theorem 3 and Theorem 4, the lemma follows. □

Note that the dominant term in the E2E jitter bound in Lemma 7 is B/r. For flows with small burst sizes, N-SCORE can guarantee tight jitter bounds.

The architecture of N-SCORE constrains jitter by elevating the lower latency bound via non-work-conserving behavior. Nevertheless, this performance trade-off is a one-off effect, as both the maximum latency and jitter bounds of N-SCORE remain inversely proportional to the service rate and proportional to the burst size. Therefore, smaller latency and tighter jitter bounds can be achieved to satisfy QoS guarantees by scaling up the service rate or diminishing the burst size.

3.3 Buffer occupancy upper bound

Based on the above observations, we can also conclude that the per-hop buffer occupancy is upper bounded as follows.

Theorem 5 *Per-hop buffer occupancy is upper bounded by* $L_h^{max} + L_{h-1}^{max} + 3NL^+$, *where* N *is the number of flows and* L^+ *is the maximum packet length among all the high priority flows.*

Proof From Lemma 1, the service intervals of consecutive packets from a flow do not overlap. In other words, $(E_h(p-1),\ F_h(p-1)]$ and $(E_h(p),\ F_h(p)]$ do not overlap. Thus, $E_h(p) \geq F_h(p-1)$.

By definition of N-SCORE, a packet service starts not earlier than its eligible time. In other words, $C_h(p) \geq E_h(p)$. Therefore, $C_h(p) \geq E_h(p) \geq F_{h-1}(p)$. On the other hand, from the definition of $\delta_{h-1}(p),\ C_{h-1}(p) + \delta_{h-1}(p) = A_h(p)$.

The packet stays in node h, during the interval $(A_h(p),\ C_h(p)]$. This interval is contained within the "staying interval" $(E_{h-1}(p)+\delta_{h-1}(p),\ F_h(p)+\frac{L^{max}}{R}]$, because $E_{h-1}(p) + \delta_{h-1}(p) \leq C_{h-1}(p) + \delta_{h-1}(p) = A_h(p)$ and $C_h(p) \leq F_h(p) + \frac{L^{max}}{R}$ from Lemma 5. From (7) and (9), $E_{h-1}(p) = F_h(p) - d_{h-1}(p) - \frac{L(p)}{r} = F_h(p) - \frac{L_{h-1}^{max}}{R_{h-1}} - \frac{L}{r} - \delta_{h-1}(p) - \frac{L(p)}{r}$.

Thus, the staying interval

$$(E_{h-1}(p) + \delta_{h-1}(p),\ F_h(p) + \frac{L_h^{max}}{R_h}]$$
$$= (F_h(p) - \frac{L_{h-1}^{max}}{R_{h-1}} - \frac{L}{r} - \frac{L(p)}{r},\ F_h(p) + \frac{L_h^{max}}{R_h}].$$

Let's say at time t, there are consecutive packets $p_1, p_2, \ldots\ p_k$ from a flow in the node h. p_1 arrived earliest. The earliest packet p_1 should meet $t < F_h(p_1) + \frac{L_h^{max}}{R_h}$. Similarly, the latest packet p_k should meet $F_h(p_k) - \frac{L_{h-1}^{max}}{R_{h-1}} - \frac{L}{r} - \frac{L(p_k)}{r} < t$. From the two inequalities, $F_h(p_k) - \frac{L_{h-1}^{max}}{R_{h-1}} - \frac{L}{r} - \frac{L(p_k)}{r} < F_h(p_1) + \frac{L_h^{max}}{R_h}$. Or we can say $F_h(p_k) - F_h(p_1) < \frac{L_h^{max}}{R_h} + \frac{L_{h-1}^{max}}{R_{h-1}} + \frac{L}{r} + \frac{L(p_k)}{r}$.

From (6), (7) and (8), $F_0(p) = \max\{F_0(p-1),\ A_0(p)\} + \frac{L(p)}{r}$ and $F_h(p) = F_{h-1}(p) + d_{h-1}(p)$, for $h > 0$. Note that $d_{h-1}(p)$ is constant over any p. These equations give $F_h(p) \geq F_h(p-1) + \frac{L(p)}{r}$, for any $h \geq 0$. Therefore, $F_h(p_k) \geq F_h(p_{k-1}) + \frac{L(p_k)}{r} = F_h(p_{k-2}) + \frac{L(p_{k-1})}{r} + \frac{L(p_k)}{r} = F_h(p_1) + \sum_{p=p_2}^{p_k} \frac{L(p)}{r}$.

We are interested in $Q_f(t) = \sum_{p=p_1}^{p_k}\{L(p)\}$, which is the total amount of traffic from the flow f in the node at time t.

From the above inequality, $\sum_{p=p_2}^{p_k} \frac{L(p)}{r} \leq F_h(p_k) - F_h(p_1) < \frac{L_h^{max}}{R_h} + \frac{L_{h-1}^{max}}{R_{h-1}} + \frac{L}{r} + \frac{L(p_k)}{r}$.

Or $Q_f(t) = \sum_{p=p_1}^{p_k} L(p) \leq r\{\frac{L_h^{max}}{R_h} + \frac{L_{h-1}^{max}}{R_{h-1}} + \frac{L}{r} + \frac{L(p_k)}{r} + \frac{L(p_1)}{r}\}$.

Thus, $Q_f(t) \leq r\{\frac{L_h^{max}}{R_h} + \frac{L_{h-1}^{max}}{R_{h-1}} + 3\frac{L}{r}\}$. The buffer occupancy bound for a flow is $r\{\frac{L_h^{max}}{R_h} + \frac{L_{h-1}^{max}}{R_{h-1}} + 3\frac{L}{r}\}$.

Now the total buffer occupancy is the sum of the flow occupancies. It is bounded as follows.

$$\begin{aligned}\sum_f Q_f(t) &\leq \sum_f r_f\{\frac{L_h^{max}}{R_h} + \frac{L_{h-1}^{max}}{R_{h-1}} + 3\frac{L_f}{r_f}\} \\ &= \sum_f r_f\{\frac{L_h^{max}}{R_h} + \frac{L_{h-1}^{max}}{R_{h-1}}\} + \sum_f r_f\{3\frac{L_f}{r_f}\} \\ &\leq L_h^{max} + L_{h-1}^{max} + 3NL^+,\end{aligned}$$

because $\sum_f r_f \leq R_h and \sum_f r_f \leq R_{h-1}$ by the admission condition. r_f and L_f are the service rate and the max packet length of flow f, respectively. N represents the total number of flows. L^+ is the maximum packet length over all the high priority flows. The theorem follows. □

Notably, the dominant term of the buffer occupancy bound is $3NL^+$, representing a per-flow overhead of approximately three packets. This modest requirement significantly alleviates the implementation complexity typically associated with PIFO (Push-In First-Out) queuing.

4 Validation

In this section, we validate whether the approach proposed in Section 3 operates as intended through simulation.

4.1 Considerations for implementation

A core node in N-SCORE is required to sort the packets according to the eligible time, put them to the output port scheduler when eligible, sort them again according to the finish time, and send them to the next node as soon as the link is available. Two different sorting mechanisms can be implemented independently, e.g. in the input port module and the output port module, as illustrated in Figure 4. By putting them in different modules, the complexity of sorting packets can be mitigated. Note that they can also be put into the output port module together.

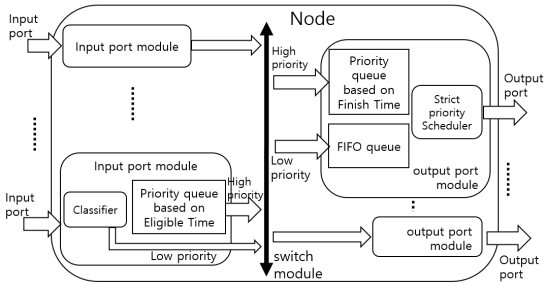


Fig. 4 An implementation practice for a core node with N-SCORE

The flows that require deterministic service are classified as high priority traffic. Other best-effort flows are considered as low priority traffic. These two priority traffic can be separated at the input port module, treated differently in the switch module, and scheduled with a strict priority scheduler in the output port module, as depicted in Figure 4. In this case with the low priority traffic, the analysis in Section 3 should be extended such that the packets from low priority traffic should be taken into account when the maximum packet length in a node h, L_h^{max}, is determined.

The E2E latency bounds given in Theorem 3 and Theorem 4 includes the sum of propagation delays from node 0 to h. There are time differences between nodes, including the differences due to the propagation delays and due to the clock mismatches. This time difference, $\delta_h(p)$, can be defined as the difference between the service completion time measured at the upstream node and the arrival time measured at the current node. The time difference can be updated only once in a while, and signaled out-of-band.

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 ETs and 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. This mechanism for the time difference compensation is also applicable for C-SCORE and described in ITU-T Y.3129 [12].

4.2 Simulation

The simulations use SimPy, a process-based discrete-event simulation framework based on standard Python, which provides various types of resources [24].

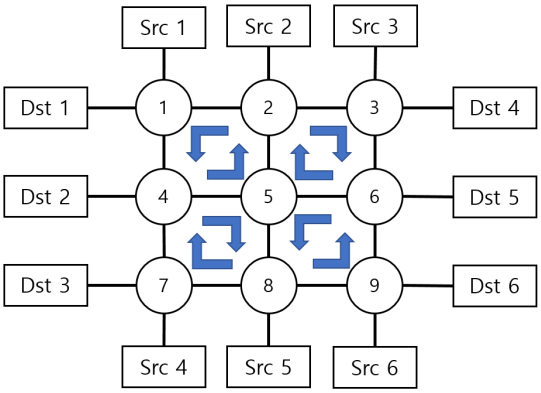


Fig. 5 Network topology used in the simulations

The network topology used in the simulation is shown in Figure 5. It represents a general network of partial mesh or grid topology, without considering a specific use case. A partial mesh is a common topology that can be seen in many real deployments, including datacenter networks. In Figure 5, arrowed links indicate the directions to follow for any traffic route. For example, from Node 2, only Node 1 and Node 3 are the next possible route. The maximum path length is 7 hops, and every source node has at least one path of this maximum length. The capacity of every link in the topology is 1Gbps. While real deployments easily exceed 1Gbps link capacity, this network represents a rather scaled down example in terms of the link capacity and the number of nodes, in order to make the network be more utilized with fewer flows. The simulation can be scaled up, but it would make the simulation much longer. The CQF, Enhanced CQF (ECQF) [25], ATS with ECQF, C-SCORE, CJVC, along with N-SCORE are compared in the simulations.

A number of studies regarding in-vehicle networks (IVNs) suggest three types of flows that request deterministic services [26–31]. These flow types and their characteristics are summarized in Table 2.

Table 2 Flow types and their characteristics in In-Vehicle Networks

Characteristic	Flow type		
	Audio	Video	Command and Control (C&C)
Maximum burst size	2 Kbit	360 Kbit	8 Kbit
Maximum packet length	2 Kbit	12 Kbit	8 Kbit
Arrival rate	200 Kbps	10.91 Mbps	800 Kbps
Burst period	10 ms	33 ms	10 ms
Deadline	10 ms	10 ms	10 ms

For Video flows under N-SCORE, aligning the service rate with the arrival rate yields a latency that exceeds 10 ms from B/r alone, thereby violating the deadline. However, based on the latency upper bound formulation of N-SCORE, the deadline requirement can be satisfied by tuning the service rate, burst size, or packet length. In the case of scaling up the service rate, the minimum rate necessary to guarantee the deadline can be mathematically determined via the latency upper bound expression in Theorem 3. Under the assumption of an ideal environment with zero time difference and the maximum hop count H of 7, the target service rate was derived as follows.

$$0.01 \geq (360K - 12K)/r + 7 \times (12K/r + 12K/1G)$$

$$0.01 \geq 432K/r + 0.000084$$

$$r \geq 432K/0.0916 \approx 47.162M$$

The deadline requirement is satisfied provided that the service rate is at least approximately 47.162 Mbps.

Alternatively, if the burst size is diminished, the deadline can be guaranteed with a significantly lower service rate. Nevertheless, this scenario assumes a modification to the flow's inherent traffic specification (Tspec). If the burst size is restricted via shaping, the deadline must be met by factoring in the shaping delay, thereby necessitating an equivalent increase in the service rate. While reducing the packet length can mitigate latency, it remains a less sensitive parameter than the service rate and thus cannot independently satisfy the deadline.

As a baseline, CQF dictates that data accumulated during a slot must be forwarded within a single slot duration, making it unsuited for Video flows characterized by large burstiness. To overcome this limitation, a shaping process is integrated to improve CQF performance. The shaping for the Video flow is implemented in accordance with [30]. In this shaping framework, the packet inter-arrival time of the video flow is restricted

to 125 us, and the maximum shaping delay is bounded at 8 ms. Consequently, the packet count scales from 30 to 64, while the packet size is halved from 12 Kbit to 6 Kbit. Due to the augmented impact of header overhead associated with a higher packet count, the total number of packets increases slightly beyond a factor of two, settling at 64.

Moreover, given that CQF yields optimal performance under periodic traffic generation, the traffic profiles of Audio and C&C flows were adjusted to exhibit strictly periodic behavior. Their periods were chosen as multiples of the video flow's inter-arrival time (125 us) to streamline the least common multiple (LCM) computation. Nonetheless, simply maintaining their low baseline arrival rates would expand the LCM of the inter-arrival times, which degrades CQF efficiency. To circumvent this, the inter-arrival times were configured at 250 us for the Audio flow and 500 us for the C&C flow, with the corresponding arrival rates increased to 8 Mbps and 16 Mbps.

The final flow characteristics used in the simulation are summarized in Table 3.

Flows A, B, and C refer to the characteristics of Audio, Video, and C&C flows, respectively. One instance of flow A and B was generated for each destination, while two instances of flow C were generated per destination. A total of 48 flows were transmitted across the topology, resulting in a bottleneck link utilization of 31.2%.

Table 3 Flow type used in the simulation

	Flow type		
Characteristic	A	B	C
Maximum burst size	2 Kbit	384 Kbit	8 Kbit
Maximum packet length	2 Kbit	6 Kbit	8 Kbit
Arrival rate	8 Mbps	11.64 Mbps	16 Mbps
Service rate	8 Mbps	48 Mbps	16 Mbps
Burst period	250 us	33 ms	500 us
Inter arrival time	250us	125us	500 us
Deadline	10 ms	10 ms	10 ms
Destination	Dst 1,6	Dst 3,4	Dst 2,5

The path of the flow is determined primarily to be the shortest path. If multiple shortest paths exist, the flow is routed through the node closer to the destination. If there are multiple nodes with the same distance to the destination, the flow is routed through the node closer to the source. We use these routing rules to ensure that each flow has a unique path. In our simulations, we only observe the flows that traveled the maximum distance for each flow type. The longest paths by flow type are shown in Table 4, with each flow type traveling a maximum distance of 7 hops.

Table 4 Longest path of each flow type in the simulation

Flow type	Longest path
A	Src5-8-7-4-5-2-1-Dst1
	Src2-2-3-6-5-8-9-Dst6
B	Src5-8-9-6-5-2-3-Dst4
	Src2-2-1-4-5-8-7-Dst3
C	Src3-3-6-5-2-1-4-Dst2
	Src6-9-6-5-8-7-4-Dst2
	Src1-1-4-5-2-3-6-Dst5
	Src4-7-4-5-8-9-6-Dst5

Packets were generated based on the burst period and inter-arrival time, with the inter-arrival time allowed to increase by up to 1.1x to observe the effects of period misalignment. Each flow generated 1,000 packets, resulting in a total of 48,000 packets. The simulation was repeated 100 times. E2E latency was observed per packet, while jitter was observed per simulation. E2E latency was measured over the longest path, and jitter was calculated as the difference between the maximum and minimum latency of a flow within a single simulation.

Comparison groups included CQF, ECQF, ATS with ECQF, CJVC, and C-SCORE. Sources 1-6 behave as the entrance nodes and the rest are the core nodes. The entrance nodes implement per-flow FIFO queues at the output port. In the core nodes, a C-SCORE node is implemented with a Push-In First-Out (PIFO) queue per input port at the output port to sort packets in the FT order. PIFO is a well-known mechanism to implement a priority queue. An N-SCORE node adds a per-input port PIFO queue that sorts packets in the ET order to put them into the PIFO queue. A CJVC node implements a per-flow FIFO queue at the output port. ECQF was implemented using TAS. The gate control list for ECQF and ATS with ECQF shown in Table 5 utilized a slot length of 125 us. For CQF, the slot length was set to 500 us so that its GCL length was implemented identically to that of ECQF.

Table 5 Gate control list for Multi-CQF and ATS with Multi-CQF

	Q6	Q5	Q4	Q3	Q2	Q1
t1	1	0	1	0	1	0
t2	1	0	1	0	0	1
t3	1	0	0	1	1	0
t4	1	0	0	1	0	1
t5	0	1	1	0	1	0
t6	0	1	1	0	0	1
t7	0	1	0	1	1	0
t8	0	1	0	1	0	1

Queues Q1–Q2 were assigned to flow B, Q3–Q4 to flow A, and Q5–Q6 to flow C, with Q6 having the highest priority. Guard bands were set to prevent packets from switching queues due to transmission delays, though inter-frame gaps were not considered. Transmission periods for flows A, B, and C were 250 us, 125 us, and 500 us, respectively. ATS with Multi-CQF added an ATS module to the output port, implemented with per-input port FIFO queues and per-flow token buckets. The token bucket conforms to the flow's Tspec, where the token generation rate is set to the service rate.

Figure 6 illustrates the distribution of E2E latency under 31.2% utilization. Flow B exhibits a broader distribution compared to Flows A and C, which is attributed to the shaping delay; as described earlier, Flow B experiences a shaping delay of up to 8 ms. The CQF-based scheduler presents latencies proportional to the slot length. Because CQF utilizes a slot length of 500 us, it fails to meet the upper bound on latency for Flow

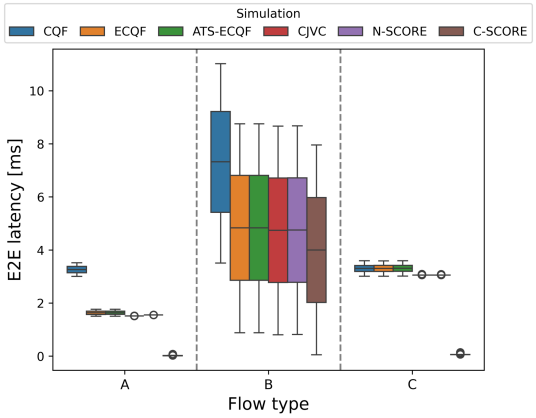


Fig. 6 Box plot of E2E latency under 31.2% utilization. The order of the boxes matches the order in the legend.

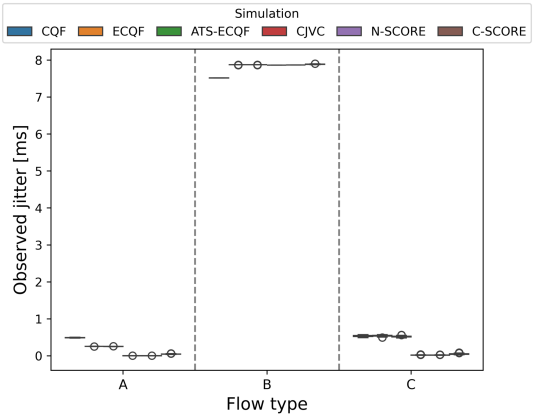


Fig. 7 Box plot of jitter under 31.2% utilization. The order of the boxes matches the order in the legend

B; furthermore, the latencies for Flows A and C are similar, as CQF does not distinguish between them. For ECQF, Flow C displays a larger latency than Flow A due to the difference in slot length, whereas no significant difference is observed when compared with ATS-ECQF, indicating that head-of-line (HOL) blocking caused by ATS has a minor impact. CJVC and N-SCORE demonstrate similar performance because they share the same theoretical upper bound. Across all flows, CJVC and N-SCORE yield lower maximum and average latencies compared to ECQF-based schedulers, which stems from the additional latency introduced by slots in ECQF. Finally, C-SCORE achieves the lowest latency across all flows, as it operates as a work-conserving scheduler.

Figure 7 illustrates the jitter distribution under 31.2% utilization. For Flow B, all schedulers exhibit a jitter of approximately 8 ms, as jitter induced by shaping delay is reflected. However, CQF demonstrates a jitter level of around 7.5 ms, owing to the increase in minimum latency at the source by an amount equivalent to the slot length. For Flows A and C, CQF-based schedulers display jitter corresponding to their slot lengths. Consequently, for Flow C, both CQF and ECQF show a jitter on the order of 500 us, whereas for Flow A, CQF and ECQF present jitter levels of around 500 us and 250 us, respectively. CJVC and N-SCORE achieve the lowest jitter in Flows A and C. Although C-SCORE exhibits higher jitter than CJVC and N-SCORE due to its work-conserving nature, it still yields lower jitter than CQF-based schedulers owing to its overall lower E2E latency.

To observe performance changes in a more congested environment, simulations were conducted with increased utilization. The flow count was tripled to achieve 93.6% utilization, with three instances for flows A and B and six instances for flow C per destination. This resulted in a total of 144,000 packets.

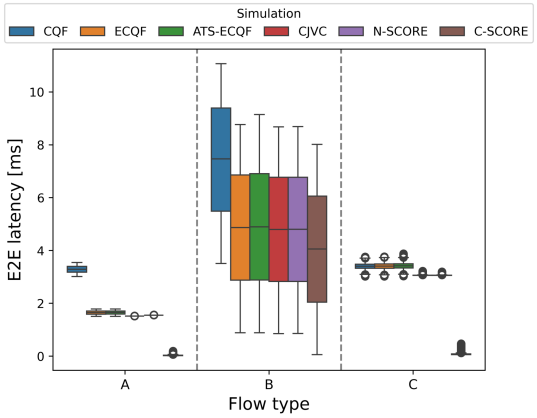


Fig. 8 Box plot of E2E latency under 93.6% utilization. The order of the boxes matches the order in the legend.

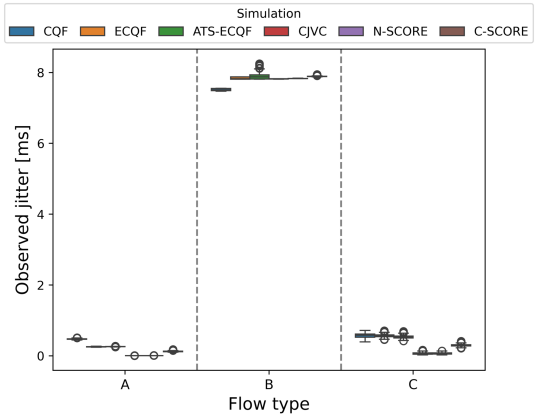


Fig. 9 Box plot of jitter under 93.6% utilization. The order of the boxes matches the order in the legend.

Figure 8 shows the E2E latency distribution under 93.6% utilization. Although the E2E latency increases with higher utilization, the magnitude of the increase remains marginal; thus, all schedulers except CQF satisfy the deadline requirements. This is because the theoretical upper bound on latency for these schedulers is independent of utilization, remaining unchanged. However, ATS-ECQF exhibits a significant latency increase for Flow B, which is attributed to HOL blocking resulting from the larger volume of packets transmitted within a single slot.

Figure 9 shows the jitter distribution under 93.6% utilization. As utilization increases, jitter also rises due to the congestion. However, compared to the results under low-utilization conditions, the jitter remains at a similar level and exhibits the same trend. The CQF-based scheduler displays jitter proportional to the slot length, CJVC and N-SCORE achieve minimal jitter levels, and C-SCORE presents an intermediate level of jitter. Notably, for Flow B, ATS-ECQF experiences an increase in jitter caused by the elevated latency from HOL blocking.

The simulation results confirm that N-SCORE provides the lowest E2E jitter bound compared to CQF-based schedulers or C-SCORE, while guaranteeing the same E2E latency upper bound as CJVC and C-SCORE.

Table 6 Number of incoming flows and theoretical buffer occupancy upper bound per node with the N-SCORE at 93.6% utilization

Node ID	Number of flow A	Number of flow B	Number of flow C	Thoretical bound [bit]
1,9	21	9	24	879,744
2,8	18	18	24	1,030,272
3,7	9	21	24	1,031,424
4,6	12	12	48	1,463,040
5	24	24	24	1,179,648

Furthermore, buffer occupancy was evaluated under the 93.6% utilization scenario with the

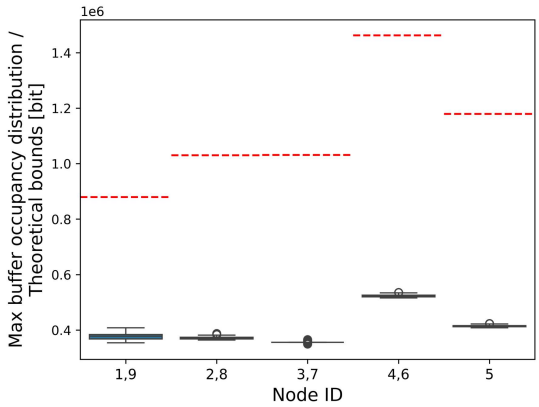


Fig. 10 Distribution of maximum buffer occupancy and theoretical upper bound per node with the N-SCORE under 93.6% utilization.

N-SCORE. We observed the maximum buffer occupancy for each node; notably, nodes positioned symmetrically with respect to node 5 were grouped together, as they share identical incoming flow configurations. Table 6 summarizes the number of incoming flows and the theoretical maximum buffer occupancy for each node. Figure 10 illustrates the distribution of maximum buffer occupancy and compares these values with their respective theoretical upper bounds. Across all simulations and nodes, the observed buffer occupancy remained consistently lower than the theoretical upper bounds.

5 Conclusion

In this paper, we proposed a non-work conserving scheduler, N-SCORE, which approximates the well-known fair queueing scheduler such as PGPS and WFQ. N-SCORE and its prequel, which is called C-SCORE, have much less complexity to be implemented in core nodes, than regular fair queuing schedulers, because the flow state information is not required to be maintained. The stateful nature of original fair queuing algorithms renders them computationally expensive and unfeasible for line-rate processing in large-scale networks. Instead, in N-SCORE and C-SCORE, the flow state information, called the finish time (FT), is carried in packets as metadata. Packets in the queue are served in the ascending order of FT. The key differentiating factor for N-SCORE from C-SCORE is its non-work conserving nature. In N-SCORE, the eligible time (ET) for a packet to be served is enforced, thus the jitter is minimized. ET is also carried in packets as metadata. These metadata are updated in nodes as the packet travels. N-SCORE was adopted as a working group draft in IETF DetNet WG recently [32].

We proved that N-SCORE has the E2E latency upper bound that is identical to both the stateful fair queuing schedulers and C-SCORE. We also proved that N-SCORE provides the E2E latency lower bound. This lower bound makes the E2E jitter bound to be much less than those of other fair queuing techniques, including another non-work conserving fair queuing scheduler, CJVC. Furthermore, we characterized the nodal buffer requirements, establishing an upper bound of approximately 3NL, where N denotes the flow

count and L the maximum packet size. This modest requirement significantly alleviates the implementation complexity typically associated with priority queuing, such as PIFO queuing.

These theoretical findings were confirmed through the simulations of a moderate sized network. Simulations were conducted using three distinct flow types under both low and high utilization environments. The results demonstrate that the proposed method exhibits significantly lower jitter compared to CQF-based schedulers, and its E2E latency can be further reduced by adjusting the service rate. Furthermore, across all evaluated scenarios, it achieved the lowest jitter levels alongside CJVC. In terms of buffer occupancy, simulations confirmed that the results did not exceed the theoretical upper bound of N-SCORE.

A large burst from a flow still remains to be a burden in controlling the jitter, even with N-SCORE. The first and the last packet of a burst, which arrive to a network almost simultaneously, have to have quite different E2E latency values with the current approaches. This fact makes the jitter of a flow with a large burst become large. For future work, we suggest reducing the jitter by treating the packets in a burst as a single entity, just as a "flowlet" in data center networks.

References

[1] International Telecommunication Union: ITU-T Y.3000-series – Network 2030 services: Capabilities, performance and design of new communication services for the Network 2030 applications. ITU-T Y Series Supplement 66, International Telecommunication Union (ITU), Geneva, Switzerland (July 2020). https://handle.itu.int/11.1002/1000/14385

[2] IEEE 802.1: Time-Sensitive Networking Task Group Home Page. Accessed: Feb. 22, 2026.

[3] Messenger, J.L.: Time-sensitive networking: An introduction. IEEE Communications Standards Magazine **2**(2), 29–33 (2018) https://doi.org/10.1109/MCOMSTD.2018.1700047

[4] IETF: Deterministic Networking (detnet) Working Group Home page. Accessed: Feb. 22, 2026.

[5] Black, D.L., Wang, Z., Carlson, M.A., Weiss, W., Davies, E.B., Blake, S.L.: An Architecture for Differentiated Services. RFC Editor (1998). https://doi.org/10.17487/RFC2475 . https://www.rfc-editor.org/info/rfc2475

[6] Thomas, L., Le Boudec, J.-Y., Mifdaoui, A.: On cyclic dependencies and regulators in time-sensitive networks. In: 2019 IEEE Real-Time Systems Symposium (RTSS),

pp. 299–311 (2019). https://doi.org/10.1109/RTSS46320.2019.00035

[7] Parekh, A.K., Gallager, R.G.: A generalized processor sharing approach to flow control in integrated services networks: the single-node case. IEEE/ACM Trans. Netw. **1**(3), 344–357 (1993) https://doi.org/10.1109/90.234856

[8] Joung, J., Kwon, J., Ryoo, J.-D., Cheung, T.: Scalable flow isolation with work conserving stateless core fair queuing for deterministic networking. IEEE Access **11**, 105225–105247 (2023) https://doi.org/10.1109/ACCESS.2023.3318479

[9] Demers, A., Keshav, S., Shenker, S.: Analysis and simulation of a fair queueing algorithm. SIGCOMM Comput. Commun. Rev. **19**(4), 1–12 (1989) https://doi.org/10.1145/75247.75248

[10] Zhang, L.: Virtual clock: a new traffic control algorithm for packet switching networks. SIGCOMM Comput. Commun. Rev. **20**(4), 19–29 (1990) https://doi.org/10.1145/99517.99525

[11] Golestani, S.J.: A self-clocked fair queueing scheme for broadband applications. In: Proceedings of INFOCOM '94 Conference on Computer Communications, pp. 636–6462 (1994). https://doi.org/10.1109/INFCOM.1994.337677

[12] International Telecommunication Union: Requirements and framework for stateless fair queuing in large scale networks including imt-2020 and beyond. ITU-T Recommendation ITU-T Recommendation Y.3129, International Telecommunication Union (ITU), Geneva, Switzerland (April 2024). https://www.itu.int/rec/T-REC-Y.3129-202404-I

[13] Joung, J., Ryoo, J.-d., Cheung, T., Li, Y., Liu, P.: Latency Guarantee with Stateless Fair Queuing. Internet-Draft draft-ietf-detnet-stateless-fair-queuing-01, Internet Engineering Task Force (July 2026). Work in Progress. https://datatracker.ietf.org/doc/draft-ietf-detnet-stateless-fair-queuing/01/

[14] Stoica, I., Zhang, H.: Providing guaranteed services without per flow management. SIGCOMM Comput. Commun. Rev. **29**(4), 81–94 (1999) https://doi.org/10.1145/316194.316208

[15] Zhang, Z.-L., Duan, Z., Hou, Y.T.: Virtual time reference system: a unifying scheduling framework for scalable support of guaranteed services. IEEE Journal on Selected Areas in Communications **18**(12), 2684–2695 (2000) https://doi.org/10.1109/49.898750

[16] Ieee standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic. IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015), 1–57 (2016) https://doi.org/10.1109/IEEESTD.2016.8613095

[17] Craciunas, S.S., Oliver, R.S.: Combined task- and network-level scheduling for distributed time-triggered systems. Real-Time Systems **52**(2), 161–200 (2016)

[18] Ieee standard for local and metropolitan area networks–bridges and bridged networks–amendment 29: Cyclic queuing and forwarding. IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017), 1–30 (2017) https://doi.org/10.1109/IEEESTD.2017.7961303

[19] Zhao, L., Pop, P., Craciunas, S.S.: Worst-case latency analysis for ieee 802.1 qbv time sensitive networks using network calculus. Ieee Access **6**, 41803–41815 (2018)

[20] Ieee standard for local and metropolitan area networks–bridges and bridged networks - amendment 34: Asynchronous traffic shaping. IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020), 1–151 (2020) https://doi.org/10.1109/IEEESTD.2020.9253013

[21] Specht, J., Samii, S.: Urgency-based scheduler for time-sensitive switched ethernet networks. In: 2016 28th Euromicro Conference on Real-Time Systems (ECRTS), pp. 75–85 (2016). https://doi.org/10.1109/ECRTS.2016.27

[22] Le Boudec, J.-Y.: A theory of traffic regulators for deterministic networks with application to interleaved regulators. IEEE/ACM Transactions on Networking **26**(6), 2721–2733 (2018) https://doi.org/10.1109/TNET.2018.2875191

[23] Goyal, P., Vin, H.M.: Generalized guaranteed rate scheduling algorithms: a framework. IEEE/ACM Transactions on Networking **5**(4), 561–571 (1997) https://doi.org/10.1109/90.649514

[24] SimPy, T.: SimPy: Discrete Event Simulation for Python. https://simpy.readthedocs.io. Accessed: Feb. 22, 2026.

[25] Li, Y.: Further details on cycle identification. Technical Report IEEE 802.1 TSN Task Group Contribution, dv-yizhou-cycle-identification-details-1122-v02, IEEE 802.1 Working Group (November 2022). https://www.ieee802.org/1/files/public/docs2022/dv-yizhou-cycle-identification-details-1122-v02.pdf

[26] NAVET, N., Migge, J.: Insights into the performance and configuration of tcp in automotive ethernet networks. In: 2018 IEEE Standards Association (IEEE-SA) Ethernet & IP@ Automotive Technology Day (2018)

[27] Mai, T.L., Navet, N., Migge, J.: A hybrid machine learning and schedulability analysis method for the verification of tsn networks. In: 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS), pp. 1–8 (2019). https://doi.org/10.1109/WFCS.2019.8757948

[28] Navet, N., Seyler, J., Migge, J.: Timing verification of realtime automotive ethernet networks: what can we expect from simulation? In: Embedded Real-Time Software and Systems (ERTS 2016) (2016)

[29] Navet, N., Mai, T.L., Migge, J.: Using machine learning to speed up the design space exploration of ethernet tsn networks (2019)

[30] Migge, J., Villanueva, J., Navet, N., Boyer, M.: Insights on the Performance and Configuration of AVB and TSN in Automotive Ethernet Networks. In: 9th European Congress on Embedded Real Time Software and Systems (ERTS 2018), Toulouse, France (2018). https://hal.science/hal-01746132

[31] Navet, N., Villanueva, J., Migge, J., Boyer, M.: Experimental assessment of qos protocols for in-car ethernet networks. In: Proc. IEEE Standards Assoc.(IEEE-SA) Ethernet IP Automot. Technol. Day, pp. 1–33 (2017)

[32] Ryoo, Y., Joung, J., CHOI, C.-h.: On-time Forwarding with Non-Work Conserving Stateless Core Fair Queuing. Internet-Draft draft-ietf-detnet-nscore-02, Internet Engineering Task Force (July 2026). Work in Progress. https://datatracker.ietf.org/doc/draft-ietf-detnet-nscore/02/

Statements and Declarations

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT) (No.RS-2026-25509411, Development of AI Data Center Interconnection technology for scaling-out AI resources).

The authors have no relevant financial or non-financial interests to disclose.

Conceptualization: Jinoo Joung, Yeoncheol Ryoo; Methodology: Jinoo Joung, Yeoncheol Ryoo, Jeong-dong Ryoo; Formal analysis and investigation: All authors; Writing - original draft preparation: Jinoo Joung, Juhyeok Kwon; Writing - review and editing: All authors; Resources: Juhyeok Kwon, Taesik Cheung; Software and Visualization: Juhyeok Kwon; Validation: Jinoo Joung, Juhyeok Kwon, Yeoncheol Ryoo; Supervision: Jinoo Joung.