IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 30 September 2022, accepted 20 October 2022, date of publication 26 October 2022, date of current version 4 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3217489

RESEARCH ARTICLE

Delay-Based Shaper With Dynamic Token Bucket Algorithm for Deterministic Networks

TATSUYA FUKUI, YUKI SAKAUE, KATSUYA MINAMI, AND TOMOHIRO TANIGUCHI
NTT Access Network Service Systems Laboratories, NTT Corporation, Tokyo 180-8585, Japan
Corresponding author: Tatsuya Fukui (tatsuya.fukui.km@hco.ntt.co.jp)

ABSTRACT To improve the experience of real-time interactive applications based on video and audio, there is a growing demand to realize deterministic networks that can transmit these media with low latency. In recent years, various deterministic network technologies have been studied such as Time Sensitive Networking (TSN). Time slot allocation type schemes such as Time-Aware Shaper (TAS) offer guaranteed delayed determinism and zero-jitter, but suffer from low network accommodation efficiency. Shaper-based techniques such as Asynchronous Traffic Shaping (ATS) provide high network accommodation efficiency and delay determinism inside the network. However, if the input traffic from the sender is bursty, End-to-End delay increases due to the shaping delay caused by the shapers at the edges of the network. In this paper, we propose Delay-Based Shaper (DBS), which dynamically controls the bandwidth while shaping the bursts so that the upper bound of the shaping delay is protected. In addition, we also propose a Dynamic Token Bucket Algorithm (DTBA) that extends the conventional token bucket algorithm to implement DBS. We show that DBS can both shape bursts and comply with the upper bound of shaping delay by comparing the behavior when bursts are input via a conventional shaper. We also demonstrate good End-to-End delay determinism and high network accommodation efficiency by applying DBS to the edge of the network.

INDEX TERMS Traffic shaping, packet delay, time-sensitive networking (TSN), deterministic networks, bounded delay.

I. INTRODUCTION

A. MOTIVATION

The COVID-19 pandemic and advances in network technologies such as 5G have increased the demand for real-time interactive applications that use video and audio media such as remote Personal Computer (PC) operations, remote robot and machine operations [1], cloud gaming [2], and remote ensembles [3]. The experience of such interactive applications is determined by the latency of the media transfer. For example, In cloud gaming, the Quality of Experience (QoE) is largely determined by the turnaround time from when the player inputs a command to when the game image rendered by the game server is displayed on the player's screen [4]. The components of the video transmission latency include device delay generated by the camera itself, propagation delay determined by the distance between transmitting/receiving points, codec delays generated by the encoding/decoding operations, and jitter delay generated in the buffer on the receiving side. The receive buffer is usually set to a very large value so that the worst of the network variation does not trigger buffer overrun. This is because, from the cost-effectiveness perspective, current real-time interactive applications are designed for Best-Effort (BE) services where Quality of Service (QoS) is not guaranteed [5]. To improve the QoE of these applications, we need deterministic network services that achieve a good delay upper bound without loss of network accommodation efficiency.

Various methods to realize deterministic networks have been proposed in the industrial domain. Time-Aware Shaper (TAS) [6] and Cyclic Queuing Forwarding (CQF) [7], which are being considered by the IEEE 802.1 the Time-Sensitive Networking (TSN) Task Group (TG), provide very strict determinism of the order of sub-microseconds for high-priority periodic traffic. However, these techniques make

The associate editor coordinating the review of this manuscript and approving it for publication was Peng-Yong Kong.

network accommodation efficiency low because each flow occupies a time slot whether it uses it or not. Furthermore, unlike traffic in industrial applications, the traffic of real-time bidirectional applications exhibit highly variable transmission cycles, and the amount of transmission data changes moment to moment, so these techniques that assume strictly periodic operations are difficult to apply as they are.

Methods are being considered to prevent traffic collisions within a First-In First-Out (FIFO) network by setting a traffic shaper at the edge of the network to reduce burstiness [8], [9]. In addition, Asynchronous Traffic Shaping (ATS), which provides a good delay upper bound by controlling the timing of transmission by testing packet eligibility at each node in the network, is being considered in TSN [10]. These technologies achieve high network accommodation efficiency because they assume packet multiplexing. However, it is known that residual traffic burstiness degrades the upper bound of delay, so fine-grained shaping needs to be provided at the edge of the network. In contrast, the video traffic generated by real-time interactive applications is bursty due to the frame behavior of the compressed video. Consequently, shaping delay, caused by shaping bursty traffic at the edge of the network, cannot be ignored. For example, if the Augmented Reality (AR) traffic model proposed in [11] sets the shaping metric at 50 Mbps, which is close to the average bit rate, a shaping delay of 40 ms or more occurs when sending 200 packets of Intra-coded Frame (I-Frame). Since the traffic pattern of video traffic can be changed by setting the resolution and video frequency in the encoder/decoder, the bit rate of the traffic shaper is difficult to optimize.

As described above, no network scheme has been proposed that combines a good End-to-End delay determinism with high network accommodation efficiency, which is necessary to improve the quality of interactive applications.

B. CONTRIBUTIONS

In this paper we propose a new shaping mechanism, Delay-Based Shaper (DBS); it reduces burstiness to within the upper boundary of the configured shaper delay requirement. In particular, we make the following contributions:

- We show the principle of DBS, which dynamically controls the rate on the basis of the number of incoming packets to satisfy the shaper delay requirement. Moreover, we propose Dynamic Token Bucket Algorithm (DTBA) extended conventional Token Bucket Algorithm to achieve this shaping operation.
- We evaluate the behavior of DBS by implementing it on the Field Programmable Gate Array (FPGA) of Layer-2 Switch (L2SW). First, we evaluate the behavior of DTBA. Second, we show that DBS can both shape bursts and comply with the upper bound of shaper delay requirement by comparing the behavior when bursts are input via a conventional Rate-Based Shaper (RBS).
- In experimental evaluations, we demonstrate that a FIFO network with DBS at the edge achieves both a good End-to-End delay determinism and high network accommodation efficiency.

C. ORGANIZATION

This article is organized as follows. Section I-D contrasts our study with related work. Section II-A introduces the concept of DBS and the difference in shaping behavior from conventional shaping. Section II-B explains the rate control approach of DBS. Section II-C details DTBA to realize DBS. Section II-D describes the configuration of deterministic networks using DBS. Section II-E shows the analysis of DBS based on network calculus. Section III-A shows the functional evaluation of DBS. Section III-B presents the network evaluation of DBS. Section IV concludes the paper.

D. RELATED WORK

As developed by IEEE TSN TG, TAS can provide transmission with minimum delay for periodic traffic by synchronizing the time of all routers/switches on the network and securing harmonious transfer by setting the Gate Control List (GCL), which specifies the transmission timing for each flow [12], [13]. However, because of its precise operation, it suffers greater drops in performance than other techniques when the traffic exceeds the design value [14] or when the GCL entries do not match the traffic pattern [15]. Therefore, in actual operation, TAS allocates time slots with some margin for high priority Scheduled Traffic (ST), resulting in lower network accommodation efficiency. Intelligent TAS [16] measures traffic and calculates the optimal time slot allocation, but it cannot handle traffic with irregular traffic cycles and data loads. Efforts have been made to increase the capacity of BE traffic by controlling the allocation time to match the actual amount of ST traffic input [17], but this does not lead to an expansion of ST capacity.

In the past, methods such as Avionic Full Duplex Switched Ethernet (AFDX), in which traffic is pre-shaped into flows at the edge of the network and multiplexed into a general FIFO network, have been considered [8], [9]. Since these methods assume packet multiplexing instead of allocating time windows, they are applicable to non-periodic traffic and have high network accommodation efficiency. The theoretical upper bounds of their delay are reported in [18] and [19]. As these studies show, high delay determinism can be obtained if the burstiness of the traffic input to a network is sufficiently reduced. However, when the input traffic is bursty, the upper bound of the delay per hop is increased, and the burstiness of the traffic is further increased. Therefore, the burstiness needs to be reduced at the edge of the network.

ATS being considered for TSN gives a better upper bound of delay than conventional FIFO networks as it performs re-shaping inside the network by the Urgency Based Scheduler (UBS) that uses Interleaved Shaping [20]. In Interleaved Shaping, eligibility checks are performed on the first packet in the queue on a per-flow basis to determine when to send the packet; this enables re-shaping on a per-flow basis without implementing a queue for each flow. However, if a bursty

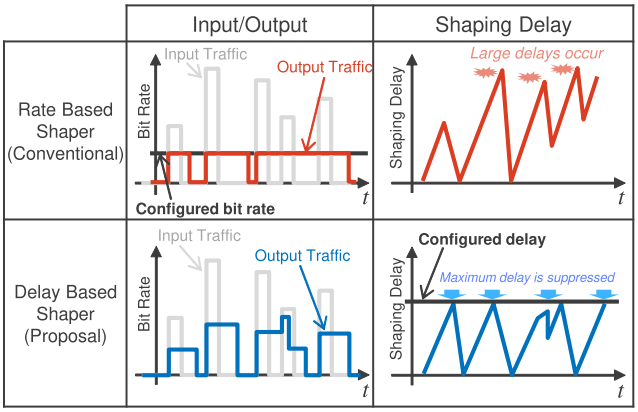


FIGURE 1. Difference in shaping behavior and shaping delay between conventional RBS and proposed DBS.

traffic flow that violates eligibility is input, a delay is imposed on the first packet in the queue, and all packets behind it are delayed. Therefore, Interleaved Shaping assumes that incoming traffic is pre-shaped to comply with the Leaky Bucket Constraint [20]. It is also shown that the upper bound of delay also depends on the burstiness of the incoming traffic [21].

A Large-scale Deterministic Network (LDN) has been proposed in which an upper bound of End-to-End delay can be provided without time synchronization or per-flow state maintenance by assigning transmission cycles that operate independently in each device of the network [22]. To prevent the transmission cycle from being occupied by a single flow, LDN is similarly configured to reduce burstiness by shaping at the edge of the NW.

To achieve a good End-to-End delay determinism, the shaping delay incurred when handling burst traffic at the edge must be considered. In [23], heuristic algorithms are proposed to design the optimal bit rate of the shaper at the edge on the basis of burst length, upper bound requirements of End-to-End delay, and network capacity. However, it is difficult for the initial settings to remain optimal given the presence of traffic with varying burst lengths and burst frequency such as video traffic.

II. DELAY-BASED SHAPER

A. CONCEPT

Fig.1 shows the operation concept of Delay-Based Shaper (DBS). A conventional shaper, called here Rate-Based Shaper (RBS), transmits packets so as to match the configured bit rate. Therefore, if the input traffic is highly bursty, packets are made to wait for long periods. Consequently, the shaper delay increases. In contrast, DBS dynamically varies the bit rate in accordance with the input bursts based on the configured shaper delay. Therefore, DBS reduces burstiness while complying with the upper bound of shaper delay.

B. RATE CONTROL APPROACH

Shaper delay d created by a shaper on the basis of the Leaky Bucket Algorithm with rate r for input of a burst of length b is determined by the following equation [24].

$$d = T + b/r \tag{1}$$

T in this equation represents the control delay of the shaper and is a constant. Thus, when a burst of length b_0 is input, the optimal output rate r_0 in DBS with the shaper delay requirement set to d_{req} is given by the following equation.

$$r_0 = b_0/(d_{req} - T) \tag{2}$$

Next, we consider the situation where a new burst with burst length b_1 is input while b_0 is being transmitted. In this case, if the output rate remains at r_0, b_1 cannot be sent within d_{req}, so the output rate is increased as follows.

$$r_1 = (b_0 + b_1)/(d_{req} - T) \tag{3}$$

After the transmission of b_0 is completed, r_1 is excessive given the requirement of d_{req},so the output rate is reduced as follows.

$$r_2 = b_1/(d_{req} - T) \tag{4}$$

Finally, to always give the optimal output rate, the output rate is set to zero after all bursts have been finished.

As described above, to implement DBS, three functions are required.

(a) Measure input burst length
(b) Increase output rate after burst input
(c) Decrease output rate after burst output

Of the above requirements, (a) can be implemented by using the metering function provided by commercial switches/routers [25]. For (b) and (c), we propose the Dynamic Token Bucket Algorithm (DTBA); it extends the conventional token bucket algorithm to enable rate control based on the shaping delay requirement. A feature of DTBA is that the number of tokens supplied varies. Each time burst arrives, DTBA updates the Token Supply Schedule (TSS), which is the future token supply plan. Also, DTBA supplies tokens in accordance with TSS. By supplying tokens for the length of the input burst within the shaper delay requirement, the burst can be assured of being released within the designated time while reducing the burstiness, regardless of the length of each packet. Furthermore, the number of states to be recorded for each DBS can be reduced because management of the burst input/output history is unnecessary.

C. DYNAMIC TOKEN BUCKET ALGORITHM

Fig.2 shows the framework of DTBA. The DTBA has four steps. First, when packets are entered, its burst length is measured and buffered. Second, the TSS array $TSS[t]$ is updated using the measured burst length and configured to meet shaping delay requirement d_{req}. Note that Measuring and Updating continue to be executed repeatedly on a cycle (e.g., 20μs) sufficiently small to detect micro-bursts. Third, tokens are supplied to the bucket in accordance with $TSS[t]$. Fourth, as in the conventional token bucket algorithm, when tokens matching the size of the packet at the head of the queue are accumulated, the packet is transmitted.

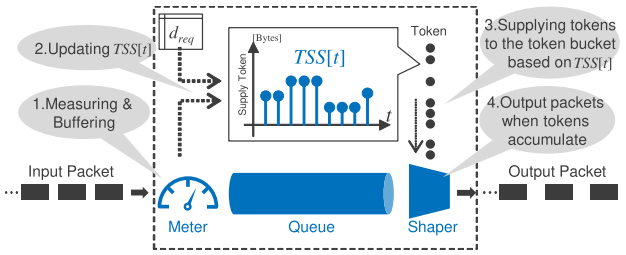


FIGURE 2. Framework of the Dynamic Token Bucket Algorithm (DTBA).

Algorithm 1 Measuring and Buffering Ingress Packets and Updating TSS

1: **Initialize:**
2: $TSS[t] = 0 \qquad (\text{all } t)$
3: **Procedure** Measuring_and_Buffering(p)
4: $b_l += p.size$
5: $Q \leftarrow p$
6: **end Procedure**
7: **Procedure** Updating_TSS()
8: $r_{add} = \dfrac{b_l}{d_{req} - (T_i + T_p)}$
9: $b_l = 0$
10: $t_{start} = t_{now} + T_p$
11: $t_{end} = t_{start} + d_{req} - (T_i + T_p)$
12: $TSS[t] += \begin{cases} r_{add} \times c & (t_{start} \le t \le t_{end}) \\ 0 & (t < t_{start}, t_{end} < t) \end{cases}$
13: **end Procedure**

Algorithm 1 details the pseudo-code for measuring and buffering ingress packets and updating TSS in DTBA. The state variables in Algorithm 1 are explained below.

1) p – a packet input to the DBS. $p.size$ indicates the packet length of p.
2) b_l – Total bytes of incoming packets from previous TSS update. This means the burst length input to the DBS in the short time between TSS updates.
3) Q – A queue of DBS. $Q.head$ indicates the packet at the head of the queue. $Q.head.size$ indicates the packet length of $Q.head$.
4) d_{req} – An upper bound of shaping delay caused by DBS. It is set from the shaping delay requirement of the flow input to the DBS.
5) T_i – Interval for TSS update. It is determined by the performance of the device on which DBS is implemented.
6) T_p – Processing delay for TSS update. This value includes the time to calculate the increase in output rate and the time to write the calculation results to memory. It is determined by the performance of the device on which DBS is implemented.
7) c – Cycle of supplying tokens in the token bucket algorithm. It is determined by the performance of the device on which DBS is implemented.

Algorithm 2 Token Supply Operation According to TSS and Packet Output

1: **Initialize:**
2: $B = 0$
3: **Procedure** Supplying_and_Output()
4: $B = B + TSS[t_{now}]$
5: **while** $Q.head.size > B$ **do**
6: output($Q.head$)
7: $B = B - Q.head.size$
8: **end while**
9: **end Procedure**

8) r_{add} – Output rate to be added in this TSS update. The number of tokens supplied at one time when the transmission rate is r is given by $r \times c$.
9) t_{now} – Current time.
10) t_{start} – Start time to add output rate in current TSS update. This is the current time t_{now} plus the processing delay T_p.
11) t_{end} – End time of additional output rate in current TSS update. Considering the processing delay involved in updating TSS and the possibility that a packet may be input immediately after the previous TSS update, this should be d_{req} minus T_i and T_p.

As shown in Line 2, TSS entries are initially all 0 to strictly increase or decrease the output rate in accordance with the burst input/output. Updating_TSS() is executed every T_i. The output rate to be added in the current TSS update is calculated by dividing the burst length b_l by the time available for transmission, which is d_{req} minus overhead time, see Line 8. Note that the above considers the processing delay and the possibility that a packet may be input immediately after the previous TSS update. Finally, the token supply value per cycle corresponding to r_{add} is added to $TSS[t]$, see Line 12. Here, the range of t to be added is limited, so the output rate is automatically reduced after this burst transmission.

Algorithm 2 details the pseudo-code of token supply operation according to TSS and packet output. In Algorithm 2, B is the amount of tokens in the token bucket. There are two differences from the conventional Token Bucket Algorithm. First, the initial state of the token bucket is empty, as shown in Line 2. As described above, since tokens are not supplied unless a packet is input, the token bucket is always empty if there is no packet in the queue of the shaper. Second, the number of tokens supplied is varied in accordance with TSS, as shown in Line 4.

Fig.3 shows an operation example of DTBA. When packets #1 and #2 are input between the TSS update timings t_0 and t_1, $TSS[t]$ is represented by the red line in this figure due to the update of t_1, and the supply of tokens starting after the period of T_p. Next, when packets #3-5 are input between t_1 and t_2, $TSS[t]$ is represented by the blue line in this figure due to the update of t_2, and the token supply speed increases after the lapse of T_p. As the token supply is continued,

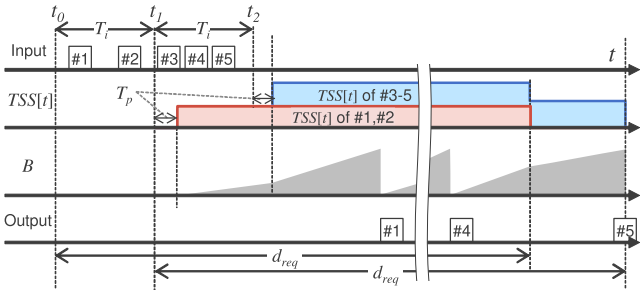


FIGURE 3. An operation example of DTBA.

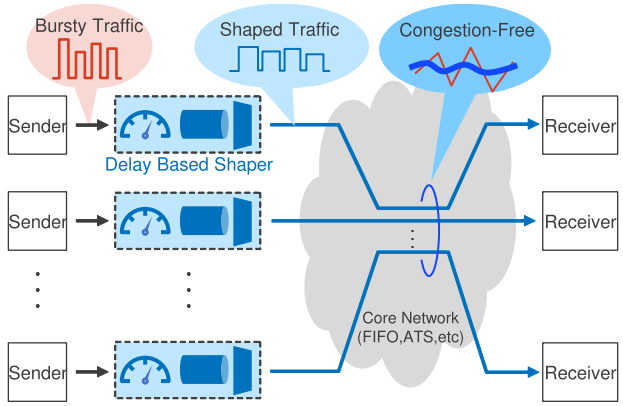


FIGURE 4. Network architecture with DBS at the edge.

packets #1, #2, and #3 are sequentially output with sufficient intervals before d_{req} elapses. After period d_{req} from t_0, the supply of tokens corresponding to packets #1 and #2 ends, so the token supply speed decreases. Finally, after period d_{req} from t_1, packet #5 is transmitted, and the supply of tokens ends.

D. APPLICATION OF DBS TO THE NETWORK

Fig.4 shows the network architecture with DBS. As with other pre-shaped schemes, placing DBS at the edge of the network reduces the burstiness of traffic from senders before they enter the core network. This prevents traffic from instantaneously fluctuating when flows are multiplexed, thus preventing congestion at routers and switches in the core network. Furthermore, unlike conventional RBS, DBS can keep the upper bound of shaping delay within a set d_{req} value, so the End-to-End delay has good determinism regardless of the burstiness of traffic in this architecture unless congestion occurs. Additionally, it can also be used in combination with ATS, LDN, etc., which are deterministic technologies inside the core network.

Note, however, that this architecture does not 'guarantee' deterministic End-to-End delay or zero jitter. The advantage of this proposal is that the increase in the End-to-End delay can be easily suppressed by simply setting the DBS to the shaping delay requirement.

Fig.5 shows the implementation patterns of DBS.

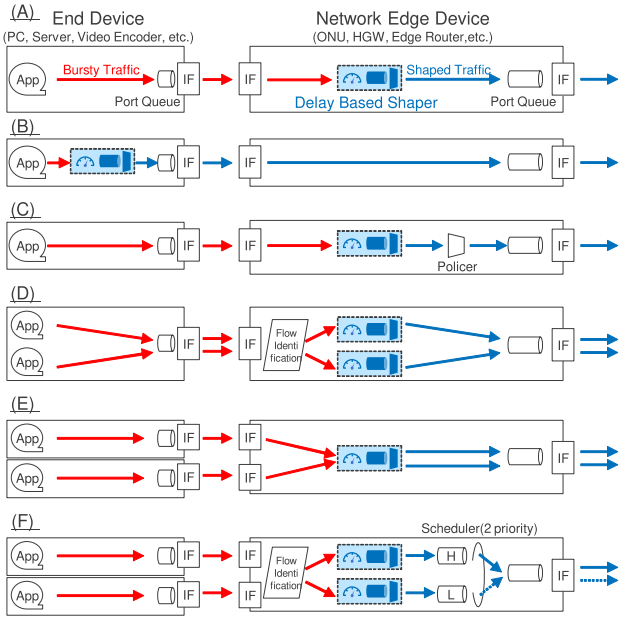


FIGURE 5. Implementation patterns of DBS.

Pattern (A) is a basic implementation. Traffic generated from an application inside the end device such as PC, a server, and a video encoder is shaped by DBS equipped in a network edge device such as Optical Network Unit (ONU), Home Gateway (HGW), and an edge router.

Pattern (B) is an implementation in which the end device also has a DBS inside. Generally, traffic regulators at the end device use the same mechanism as conventional RBS. However, in this case, as discussed previously, the regulation delay increases when the burstiness of the traffic generated by the application is large. Therefore, the DBS is applied as a traffic regulator in this implementation.

Pattern (C) is an implementation in which a policer is applied at the back end of DBS. DBS by itself does not have a mechanism to limit the output rate limit, so the output rate will be higher when there is a greedy/selfish talker. To protect the network in operation, a policer should be used to limit the amount of traffic entering the network.

Patterns (D) and (E) show implementation variations of handling multiple flows. In Pattern (D), two flows from the same interface are input to separate DBSs. The advantage of this implementation is that each flow can be shaped with different delay requirements, but the disadvantage is that it consumes a lot of DBS resources. In Pattern (E), two flows from different IFs are input to the same DBS. Since the DBS maintains an upper bound of the shaping delay regardless of the behavior of the input traffic, there is no problem with inputting the two flows into the same DBS.

Pattern (F) shows how to apply to multi-class flows. Since DBS itself does not have the behavior to handle multi-class flows, each multi-class flow should be applied to a separate DBS and a priority control should be implemented by the

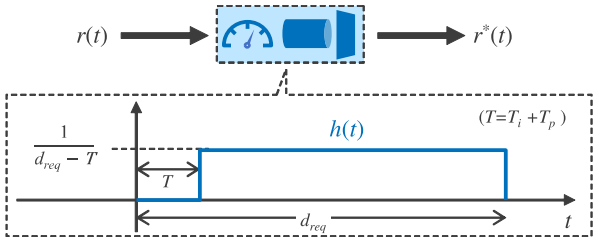


FIGURE 6. Formulation of DBS.

scheduler (e.g. Strict Priority) at a later stage. The advantage of applying DBS over conventional priority control is that it reduces waiting time of low priority traffic. Normally, while a burst of high priority traffic is being transmitted, low priority traffic does not get a chance to transmit. However, by applying DBS, bursts of high priority traffic are reduced, so low priority traffic can be transmitted in between packets.

E. ANALYSIS OF DBS BASED ON NETWORK CALCULUS

In this section, we analyze DBS on the basis of network calculus. Because it is computationally simpler to consider continuous time, we analyze using a fluid model [24].

1) THE FORMULATION OF DBS

The formulation of DBS is shown in Fig.6. Observing the token supply operation, the DBS can be interpreted as responding to the input data by stretching the input data with $d_{req} - T$ after T, where T is $T_i + T_p$. Therefore, the output rate function $r^*(t)$ of DBS can be obtained as a result of the convolution integral of the response function $h(t)$ given by the following equation with the input rate function $r(t)$.

$$h(t) = \begin{cases} \dfrac{1}{d_{req} - T} & (T \le t \le d_{req}) \\ 0 & (t < T, d_{req} < t) \end{cases} \tag{5}$$

$R(t)$ is the input cumulative function of DBS and $R^*(t)$ is the output cumulative function of DBS. Noting $R(t) = \int_0^t r(s)ds$, $R(0) = 0$ and $r(t) \ge 0$, $r^*(t)$ is obtained as.

$$\begin{aligned} r^*(t) &= \int_{-\infty}^{\infty} r(\tau)h(t-\tau)d\tau \\ &= \int_{-\infty}^{\infty} h(\tau)r(t-\tau)d\tau \\ &= \frac{1}{d_{req} - T}\int_{T}^{d_{req}} r(t-\tau)d\tau \\ &= \frac{1}{d_{req} - T}\left(R(t-T) - R(t-d_{req})\right) \end{aligned} \tag{6}$$

Similarly, since $R^*(t) = \int_0^t r^*(s)ds$, $R^*(t)$ is expressed by the following equation.

$$R^*(t) = \frac{1}{d_{req} - T}\left(\int_0^t R(s-T) - R(s-d_{req})ds\right) \tag{7}$$

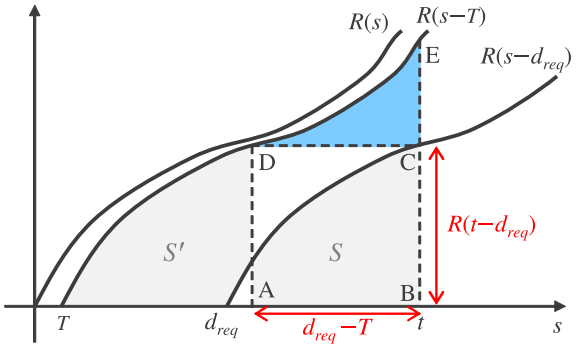


FIGURE 7. Illustration of $R(s-T)$ and $R(s-d_{req})$ in equation (7).

2) UPPER BOUND OF THE SHAPING DELAY CAUSED BY DBS

To show that the upper bound of the shaping delay caused by DBS is d_{req}, we need to prove that the following equation holds in $R^*(t)$ [24].

$$R^*(t) \ge R(t - d_{req}) \tag{8}$$

Noting that R is a monotonically increasing function, $R(s-T)$, and $R(s-d_{req})$ in equation (7) can be illustrated as in Fig.7. The value of the definite integral on the right side in equation (7) is the area bounded by the x-axis, $R(s-T)$, $R(s-d_{req})$, and line segment EC in Fig.7. Since S and S′ are equal, their area is the sum of the quadrilateral ABCD and the area painted in blue. Given that the area of quadrilateral ABCD is $(d_{req} - T) \times R(t - d_{req})$, $R^*(t)$ can be transformed as follows.

$$\begin{aligned} R^*(t) &= \frac{1}{d_{req} - T}\left(\int_0^t R(s-T) - R(s-d_{req})ds\right) \\ &\ge \frac{1}{d_{req} - T} \times \left(d_{req} - T\right) \times R(t - d_{req}) \\ &= R(t - d_{req}) \end{aligned} \tag{9}$$

From the derivation of equation (8), the upper bound of the shaping delay caused by DBS is shown to be d_{req}.

3) CONGESTION-FREE CONDITION WHEN MULTIPLEXING FLOWS WITH DBS

To avoid a congestion when multiplexing flows, the sum of the instantaneous output rates of each flow should not exceed the capacity of the network. The instantaneous output rate of the flow with DBS is $r^*(t)$, as shown in equation (6). $R(s-T) - R(s-d_{req})$ in this equation means the amount of data input to the DBS during $d_{req} - T$. If maximum amount of burst arriving during $d_{req} - T$ is b_{max}, the maximum instantaneous output rate of the flow with DBS r_{max} is given by the following equation.

$$r_{max} = \frac{b_{max}}{d_{req} - T} \tag{10}$$

Therefore, assuming that DBS is applied to each flow and the worst-case scenario where all bursts collide at the same time,

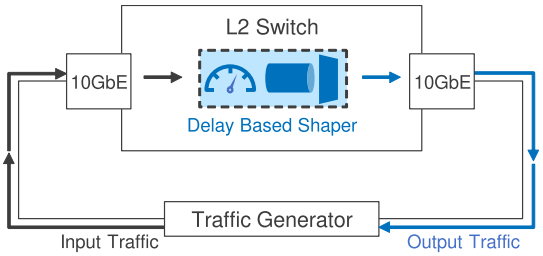


FIGURE 8. Experimental configuration used for functional evaluation.

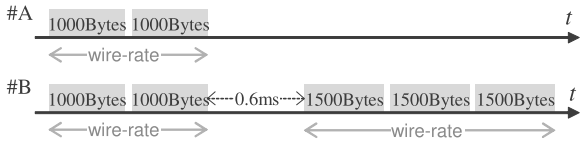


FIGURE 9. The traffic pattern used in the evaluation of token supply operation by DTBA.

the congestion-free condition when multiplexing flows with DBS is expressed in the following equation.

$$L \geq \sum_{k}^{allflow} \frac{b_{max_k}}{d_{req_k} - T_k} \tag{11}$$

In the above equation, L denotes the capacity of the bottleneck link of the network, and the subscript k denotes the parameters of the k-th flow. From the above, even considering collision scenarios, as long as the sum of the rates averaged by d_{req_k} does not exceed the amount of incoming bursts, the network accommodation efficiency can be improved while achieving good delay determinism.

III. EXPERIMENTAL EVALUATION

A. FUNCTIONAL EVALUATION

1) EXPERIMENTAL SETUP

Fig.8 shows the experimental configuration. As shown, arbitrary traffic is input to DBS via 10 Gigabit Ethernet (10GbE) by a traffic generator and output traffic is observed. DBS is implemented on a FPGA of a L2SW. T_p, T_i, and c are set to 20 μs. The operation of the token supply is evaluated by observing the transition log of the tokens stored in the token bucket.

2) TOKEN SUPPLY OPERATION BY DTBA

Fig.10 shows token bucket transitions given the input of traffic patterns shown in Fig.9 are input. In Fig.10, the vertical axis is the tokens stored in the token bucket and the horizontal axis is elapsed time since the first packet is input. In Fig.9, note that the "wire-rate" in the figure means that packets arrive consecutively at the shortest possible interval (specifically, Inter-Frame Gap is 12 bytes). The number of tokens supplied per cycle for $d_{req} = 1.0$ and 3.0 when traffic pattern #A is input is obtained by referring to Algorithm 1

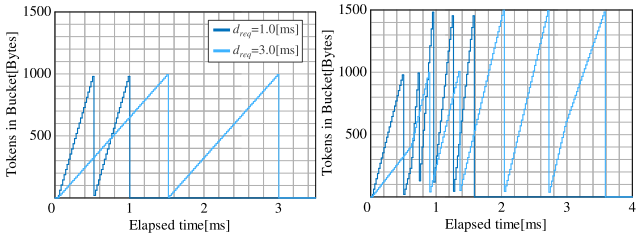


FIGURE 10. Illustration of processes of token bucket condition when traffic patterns shown Fig.9 are input.

Line 8, 12 as follows.

$$\frac{1000 \times 2}{\{1 - (0.02 + 0.02)\} \times 10^{-3}} \times 20 \times 10^{-6} \simeq 42 \tag{12}$$

$$\frac{1000 \times 2}{\{3 - (0.02 + 0.02)\} \times 10^{-3}} \times 20 \times 10^{-6} \simeq 13.5 \tag{13}$$

From Fig.10(a), the number of token supply is consistent with the above equation. Furthermore, in this figure, the decrease in the number of tokens from around 1000 bytes to around 0 bytes indicates that a packet is being sent out. Therefore it can be seen that packets are output with sufficient packet spacing within d_{req}.

As shown in Fig.10, in traffic pattern #B, where three 1500 bytes packets are input 0.6 ms after the input of traffic pattern #A, the token supply per cycle increases by the value given by the following calculation.

$$\frac{1500 \times 3}{\{1 - (0.02 + 0.02)\} \times 10^{-3}} \times 20 \times 10^{-6} \simeq 94.5 \tag{14}$$

$$\frac{1500 \times 3}{\{3 - (0.02 + 0.02)\} \times 10^{-3}} \times 20 \times 10^{-6} \simeq 30.375 \tag{15}$$

Similarly, after d_{req} elapses, when the supply of tokens for the first two 1000 bytes packets is completed, we find that the token supply speed decreases. In addition, the size of the token amount decrease confirms that two groups of packets of different sizes can be sent within d_{req} with an acceptable interval. This demonstrates that DTBA can dynamically adjust the token supply to control the timing of packet output, regardless of packet size or input pattern.

3) BURST REDUCTION OPERATION BY DBS

The difference in behavior between the proposed DBS and conventional RBS is shown by the service curve of network calculus terminology [24].

Fig.12 compares the service curve of DBS and RBS when the target shaping delay is set to 1 ms or 1.5 ms. In this evaluation, traffic patterns shown in Fig.11 are input. Committed Burst Size (CBS) in RBS is 2 KBytes.

In Fig.12(a), when the output rate is 500 Mbps in RBS, the burst length of the first burst exceeds the output rate, so that the shaping delay increases. On the other hand, when the output rate is 1000 Mbps in RBS, the shaping delay does not increase because the output rate is appropriate for the first burst. However, the second burst is sent too rapidly so

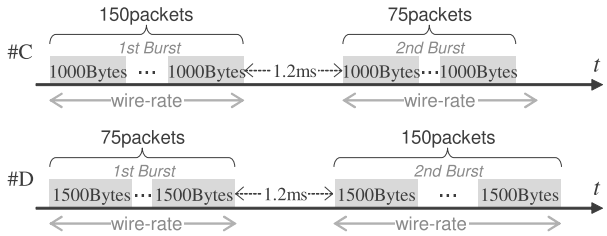
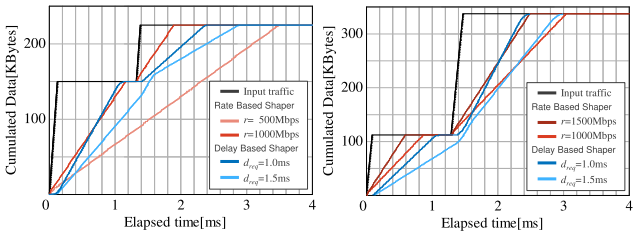


FIGURE 11. The traffic pattern used in evaluating Burst reduction operation by DBS.



(a) Traffic Pattern #C (b) Traffic Pattern #D

FIGURE 12. Comparisons between the service curve of DBS and RBS when the target shaping delay is set to 1 *ms*.

leveling is inadequate. In contrast, since the proposed DBS dynamically changes the output rate in accordance with the input burst length, the two bursts are output as uniformly as possible while keeping the shaping delay within d_{req}. In particular, in DBS with $d_{req} = 1.5$ms, even if the second burst arrives before the end of the first burst, the sending of all packets can be completed within d_{req} as the output rate is dynamically changed.

As shown in Fig.12(b), when an output rate is 1000 Mbps in RBS, the shaping delay increases with the second burst because the burst length of traffic pattern #C is larger than that of #D. However, in DBS, even if the traffic pattern changes, it can be seen that the burst is leveled within d_{req}.

As a result, in the conventional RBS, the shaping delay is uncontrollable because the output rate is constant, so the time required for transmission varies with the burst length, whereas in the proposed DBS, the output rate can change dynamically, so the shaping delay can be kept within d_{req} regardless of the burst length. In RBS, although the shaper delay can be reduced by setting a large output rate, the bursts cannot be completely leveled when the burst length of the traffic is small. In contrast, DBS makes it possible to transmit at the appropriate output rate. Furthermore, if the requested shaping delay is to be changed, DBS makes it possible to simply change the value of the shaping delay d_{req}. Therefore, DBS is useful for applications with variable burst lengths of traffic, or when each flow has different requirements.

B. NETWORK EVALUATION

1) EXPERIMENTAL SETUP

Fig.13 shows the experimental configuration of the network evaluation. As shown in this figure, 10 10GbE links are connected to L2SW #A from a traffic generator, and traffic

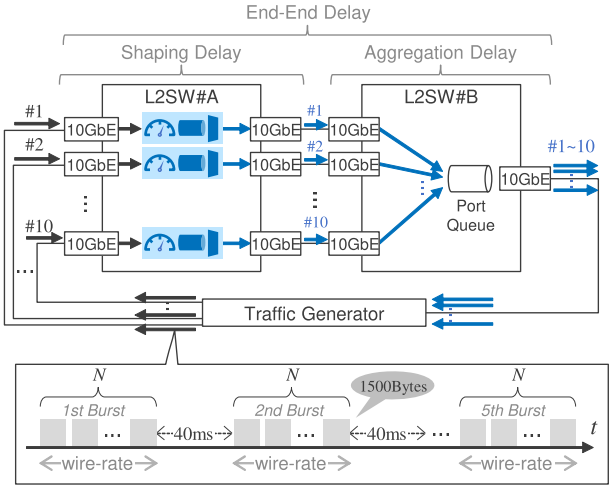


FIGURE 13. The experimental configuration for network evaluation.

TABLE 1. d_{req}[ms] for each flow used in the network evaluation.

#	Flow Set 1	Flow Set 2	Flow Set 3
1	1.0	1.0	1.0
2	2.0	1.5	3.0
3	3.0	2.0	5.0
4	4.0	2.5	7.0
5	5.0	3.0	9.0
6	6.0	3.5	11.0
7	7.0	4.0	13.0
8	8.0	4.5	15.0
9	9.0	5.0	17.0
10	10.0	5.5	19.0

flows from each link are simultaneously input; each flow consists of bursts of N packets repeated five times in a cycle of 40 ms. L2SW #A implements DBSs to shape the input bursts. These outputs from DBSs are input to L2SW #B via 10GbE links, and output from only one 10GbE link at L2SW #B. Thus, traffic collisions may cause delays at the egress port of L2SW #B. The time from the emission of the traffic by the traffic generator to the reception by the traffic generator again via L2SW #A and L2SW #B is evaluated as End-to-End Delay. The time from input to L2SW #B to output from L2SW #B is evaluated as Aggregation Delay. Note that L2SWs generate fixed delay of about 20 μs.

Table 1 shows the value of d_{req} to be set in DBS for each flow. For example, for Flow Set 1 d_{req} is increased in increments of 1 from 1.0 to 10.0. Flow Set 2 has stricter delay requirements than Flow Set 1. Flow Set 3 has looser delay requirements than Flow Set 1.

2) END-TO-END DELAY

Fig.14 shows the maximum End-to-End delay of each flow versus the input burst length. In this figure, only the plot of an odd number of Flow numbers is shown. Additionally, in

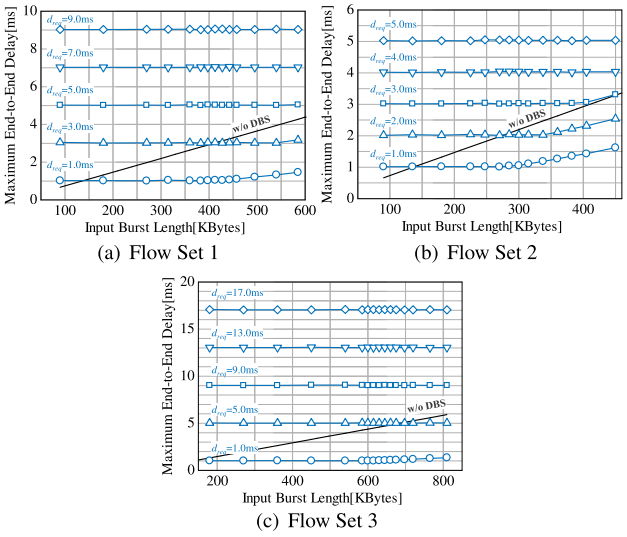


FIGURE 14. The maximum End-to-End delay of each flow versus input burst length.

this figure, the maximum End-to-End delay when simply multiplexing bursts without DBS is shown by the black line. As shown in the figure, in all Flow Sets, the maximum End-to-End delay does not increase until the burst length exceeds a certain threshold. The reason is that DBS reduces burstiness enough to prevent congestion at the aggregation point. In addition, the threshold burst length at which the End-to-End delay increases is smaller the more severe shaping delay requirements. This is because bursts are output faster when shaping delay requirements are severe, and the total bandwidth that each flow is aggregated is likely to exceed the link speed. Note that the maximum End-to-End delay of the flow with $d_{req} = 1.0$ appears to degrade first, but this reflects a quirk of the experimental method in handling the last packet of a burst with $d_{req} = 1.0$.

From the above, it is concluded that DBS can be used to realize deterministic networks by assuring an upper bound on the delay for each flow in accordance with the flow's delay requirements without having to know the characteristics of the input traffic.

3) AGGREGATION DELAY

Fig.15 shows the maximum Aggregation Delay versus the total maximum instantaneous output rates r_{sum}. The maximum Aggregation Delay shows the maximum value of Aggregation Delay measured during the experiment at the flow #1 with $d_{req} = 1.0$ and displayed on the logarithmic axis. r_{sum} is the sum of the instantaneous maximum output rates in DBS and is the right-hand side of equation (11). The reason for using r_{sum} is to evaluate how much traffic can be accommodated in the network while maintaining a good delay determinism by using DBS. r_{sum} is calculated from the experimental condition that is d_{req} for each flow, T_p, T_i and the input burst length. Thereby, r_{sum} is different for each Flow set even with the same burst size. For example, when the burst

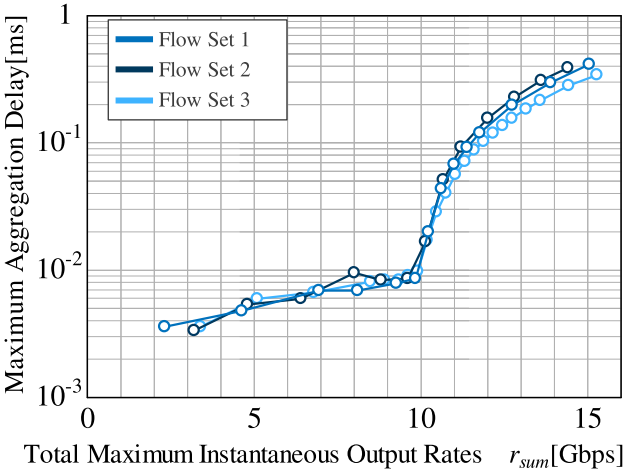


FIGURE 15. The maximum Aggregation Delay versus the total maximum instantaneous output rates r_{sum}.

length is 200 KBytes, the r_{sum} of each Flow Set is determined as follows.

$$r_{sum} = \begin{cases} \dfrac{200 \times 1000 \times 8}{\{1.0 - 0.04\} \times 10^{-3}} + \\ \dfrac{200 \times 1000 \times 8}{\{2.0 - 0.04\} \times 10^{-3}} + \cdots \\ = 5.076[\text{Gbps}] & \text{(Flow Set 1)} \\ \dfrac{200 \times 1000 \times 8}{\{1.0 - 0.04\} \times 10^{-3}} + \\ \dfrac{200 \times 1000 \times 8}{\{1.5 - 0.04\} \times 10^{-3}} + \cdots \\ = 7.019[\text{Gbps}] & \text{(Flow Set 2)} \\ \dfrac{200 \times 1000 \times 8}{\{1.0 - 0.04\} \times 10^{-3}} + \\ \dfrac{200 \times 1000 \times 8}{\{3.0 - 0.04\} \times 10^{-3}} + \cdots \\ = 3.722[\text{Gbps}] & \text{(Flow Set 3)} \end{cases} \tag{16}$$

As shown in Fig.15, regardless of the value of d_{req} set for each flow, the maximum Aggregation Delay can be kept less than 10 μs until r_{sum} exceeds the link capacity of 10 Gbps. This is because, the burst leveling provided by DBS prevents congestion from occurring until the link capacity is reached. From the above, it can be concluded that DBS can provide deterministic networks without reducing network accommodation efficiency.

IV. CONCLUSION AND FUTURE WORK

In this article, we proposed a new shaping mechanism, Delay-Based Shaper(DBS); its operation considers the requested shaping delay, unlike conventional Rate-Based Shaping. To realize DBS, we proposed the Dynamic Token Bucket Algorithm(DTBA), which extends the conventional token bucket algorithm by adopting a dynamic token supply. Experiments on DBS implemented in hardware demonstrated that DTBA could output packets within the shaping delay due

to its dynamic control of the token supply; measured results verified the ability of DBS to shape bursts regardless of the length and timing of input. Finally, through experiments that examined the input of multiple traffic flows, we have shown that DBS can realize deterministic networks able to provide a good End-to-End delay determinism even if each flow has different delay requirements without degrading network accommodation efficiency.

Future work will include examining the result of combining DBS with other technologies (ATS, LDN, etc.) and large-scale network simulation for a real network and applications.

REFERENCES

[1] X. V. Wang and L. Wang, "A literature survey of the robotic technologies during the COVID-19 pandemic," *J. Manuf. Syst.*, vol. 60, pp. 823–836, Jul. 2021.

[2] W. Cai, R. Shea, C. Y. Huang, K. T. Chen, and J. Liu, "A survey on cloud gaming: Future of computer games," *IEEE Access*, vol. 4, pp. 7605–7620, 2016.

[3] C. Rottondi, C. Chafe, C. Allocchio, and A. Sarti, "An overview on networked music performance technologies," *IEEE Access*, vol. 4, pp. 8823–8843, 2016.

[4] M. Claypool and D. Finkel, "The effects of latency on player performance in cloud-based games," in *Proc. 13th Annu. Workshop Netw. Syst. Support Games*, Dec. 2014, pp. 1–6.

[5] Y. Zhao, A. Zhou, and X. Chen, "Reducing latency in interactive live video chat using dynamic reduction factor," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, May 2020, pp. 1–6.

[6] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic*, IEEE Standard 802.1Qbv-2015, Mar. 2016, pp. 1–57.

[7] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 29: Cyclic Queuing and Forwarding*, IEEE Standard 802.1Qch-2017, Jun. 2017, pp. 1–30.

[8] J. Loeser and H. Haertig, "Low-latency hard real-time communication over switched Ethernet," in *Proc. 16th Euromicro Conf. Real-Time Syst. (ECRTS)*, 2004, pp. 13–22.

[9] X. Fan, M. Jonsson, and J. Jonsson, "Guaranteed real-time communication in packet-switched networks with FCFS queuing," *Comput. Netw.*, vol. 53, no. 3, pp. 400–417, Feb. 2009.

[10] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 34: Asynchronous Traffic Shaping*, IEEE Standard 802.1Qcr-2020, Nov. 2020, pp. 1–151.

[11] T. Tojo, H. Baba, S. Yasukawa, and Y. Okazaki, "Micro-burst analysis and mitigation for 5G low-latency communication services," in *Proc. IEEE 2nd 5G World Forum (5GWF)*, Sep. 2019, pp. 232–236.

[12] Z. Zhou, J. Lee, M. S. Berger, S. Park, and Y. Yan, "Simulating TSN traffic scheduling and shaping for future automotive Ethernet," *J. Commun. Netw.*, vol. 23, no. 1, pp. 53–62, Feb. 2021.

[13] S. Thangamuthu, N. Concer, P. J. L. Cuijpers, and J. J. Lukkien, "Analysis of Ethernet-switch traffic shapers for in-vehicle networking applications," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, 2015, pp. 55–60.

[14] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS)," *IEEE Access*, vol. 7, pp. 44165–44181, 2019.

[15] R. Nakayama and Y. Ito, "Study on affect of GCL of time-aware shaper on QoS for IEEE802.1 TSN networks," in *Proc. IEEE 9th Global Conf. Consum. Electron. (GCCE)*, Oct. 2020, pp. 946–947.

[16] N. Shibata, P. Zhu, K. Nishimura, Y. Yoshida, K. Hayashi, M. Hirota, R. Harada, K. Honda, S. Kaneko, J. Terada, and K.-I. Kitayama, "First demonstration of autonomous TSN-based beyond-best-effort networking for 5G NR fronthauls and 1,000+ massive IoT traffic," in *Proc. Eur. Conf. Opt. Commun. (ECOC)*, Dec. 2020, pp. 1–4.

[17] N. Shibata, S. Kaneko, R. Harada, K. Honda, and J. Terada, "Autonomous dynamic window shaping and rerouting for a service-converged layer-2 network with a time-aware shaper accommodating mobile fronthaul and IoT backhaul," *J. Opt. Commun. Netw.*, vol. 13, no. 5, pp. 108–115, May 2021.

[18] J. C. R. Bennett, K. Benson, A. Charny, W. F. Courtney, and J. Y. L. Boudec, "Delay jitter bounds and packet scale rate guarantee for expedited forwarding," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 529–540, Aug. 2002.

[19] M. Boyer and C. Fraboul, "Tightening end to end delay upper bound for AFDX network calculus with rate latency FIFO servers using network calculus," in *Proc. IEEE Int. Workshop Factory Commun. Syst.*, May 2008, pp. 11–20.

[20] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched Ethernet networks," in *Proc. 28th Euromicro Conf. Real-Time Syst. (ECRTS)*, Jul. 2016, pp. 75–85.

[21] J.-Y. Le Boudec, "A theory of traffic regulators for deterministic networks with application to interleaved regulators," *IEEE/ACM Trans. Netw.*, vol. 26, no. 6, pp. 2721–2733, Dec. 2018.

[22] B. Liu, S. Ren, C. Wang, V. Angilella, P. Medagliani, S. Martin, and J. Leguay, "Towards large-scale deterministic IP networks," in *Proc. IFIP Netw. Conf. (IFIP Networking)*, Jun. 2021, pp. 1–9.

[23] S. Martin, P. Medagliani, and J. Leguay, "Network slicing for deterministic latency," in *Proc. 17th Int. Conf. Netw. Service Manage. (CNSM)*, Oct. 2021, pp. 572–577.

[24] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*, vol. 2050. Berlin, Germany: Springer, 2001.

[25] T. Stevenson. *Nanosecond Buffer Visibility With Hardware-Based Microburst Detection*. Cisco. Accessed: Oct. 20, 2022. [Online]. Available: https://blogs.cisco.com/datacenter/nanosecond-buffer-visibility-with-hardware-based-microburst-detection

TATSUYA FUKUI received the B.E. and M.E. degrees from Waseda University, Japan, in 2008 and 2010, respectively. He is currently working at NTT Access Network Service Systems Laboratories. His research interest includes research and development of carrier networks, such as wide-area Ethernet systems.

YUKI SAKAUE received the B.E. and M.E. degrees in applied physics from the University of Hokkaido, Japan, in 2010 and 2012, respectively. In 2012, he joined NTT Access Network Service Systems Laboratories. His research interest includes access network systems mainly related to passive optical network systems and wireless quality control systems.

KATSUYA MINAMI received the B.E., M.E., and Ph.D. degrees from Osaka University, Japan, in 1998, 2000, and 2003, respectively. He is currently working with NTT Access Network Service Systems Laboratories. His research interests include research and development of optical access systems and wide-area Ethernet systems.

TOMOHIRO TANIGUCHI received the B.E. and M.E. degrees in precision engineering from The University of Tokyo, Japan, in 2000 and 2002, respectively, and the Ph.D. degree in electrical, electronic, and information engineering from Osaka University, Japan, in 2010. In 2002, he joined NTT Access Network Service Systems Laboratories. His research interest includes optical access systems mainly related to optical heterodyne technologies, radio-on-fiber transmission, and video distribution systems.