IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 7, 2021, accepted March 21, 2021, date of publication March 24, 2021, date of current version April 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068515

Zero Jitter for Deterministic Networks Without Time-Synchronization

JINOO JOUNG[1] AND JUHYEOK KWON[2]
[1]Department of Human-Centered Artificial Intelligence, Sangmyung University, Seoul 03016, South Korea
[2]Department of Intelligence Information Engineering, Sangmyung University, Seoul 03016, South Korea

Corresponding author: Jinoo Joung (jjoung@smu.ac.kr)

This work was supported by the Institute for Information and Communications Technology Promotion (IITP) funded by the Korea Government (MSIT) under Grant IITP-2018-0-00846.

ABSTRACT The demands for deterministic network services are getting significant. IEEE Time sensitive networking (TSN) task group has aimed for deterministic services through IEEE 802 networks. TSN's solutions rely on the time-synchronization and the slot scheduling coordinated throughout a network. The former requires costly hardware implementations for accuracy and scalability. The latter requires a schedulability analysis, which is proven to be NP-complete and not scalable. For modest-to-large scale dynamic networks, a more flexible solution is necessary. We propose a scalable framework that guarantees any desired jitter upper bound, including zero jitter. The framework is composed of; packets with relative time-stamps, a network that guarantees latency bounds, and buffers at the network egress. The latency bounds of a network can be of any value. The buffer holds packets for predetermined intervals to reproduce arbitrary inter-arrival intervals to inter-departure intervals, based on the network's latency bound. The idea behind the proposed framework is to minimize the latency first with work-conserving schedulers, and then adjust the jitter at the egress of a network. We argue that this is simpler than minimizing latency and jitter at the same time. A direct benefit of such simplicity is its scalability. We examine the network solutions that guarantee latency bound, such as the DiffServ and the Flow aggregate-interleave regulator (FAIR), as the candidates for the component in the framework. We show with a case study and simulations that the proposed framework can guarantee smaller latency bounds, with zero jitter, than those of the TSN's synchronous approach.

INDEX TERMS Deterministic network, jitter, latency, time-synchronization, TSN.

I. INTRODUCTION

A. DEMANDS FOR DETERMINISTIC NETWORKS

The demands for deterministic network services are getting significant. It is strong in closed small networks such as in-car networks or smart factory networks, in which the latency and jitter requirements are clearly defined. Typical control cycles in automated factories are 100-1000ms with a jitter of $<$1ms, while in robotics control cycles can go $<$1ms with a jitter of $<1\mu$s [1]–[3]. The On-Board Software Reference Architecture Network Communication Specification (OSRA-NET), the European standard for satellite network, defines seven communication classes and their jitter bounds as well as the latency bounds. OSRA-NET is a small network with at most 15 end-points, expected to accommodate 25 end-points in a near future [4]. See Table 1 [4].

TABLE 1. OSRA-NET traffic classes [4].

Class	Frequency (Hz)	Data rate (bps)	Max Jitter (ms)	Max Latency (ms)
1	0.1-1	100-10K	10	10
2-a	8-10	<1M	5-10	10
2-b	8-10	<1M	5-10	10
3	8-10	<250K	10	10
4	0.1-1	>100M	<100	<100
5-a	10-1K	<3M	0.5-1	0.5
5-b	10-1K	<3M	0.5-1	0.5
6	1-10	>100M	2	10
7	1-10	100-1K	1	2

A class 6 flow in OSRA-NET, for example, requires maximum latency of 10ms and maximum jitter of 2ms, while having more than 100Mbps data rate.

Common public radio interface (CPRI) in 5G network is another example of small networks with stringent latency

The associate editor coordinating the review of this manuscript and approving it for publication was Petros Nicopolitidis.

and jitter requirements. CPRI is a packet-based constant-bit-rate protocol developed to transport data between the remote radio heads (RRHs) and baseband units (BBUs) pool. CPRI network is usually a simple single-hop network. While achieving the throughput of 10Gbps as required in the fronthaul networks, CPRI is expected to maintain the maximum end-to-end latencies less than 100-250μs [5], [6], jitter within 65∼100ns [6], [7] and BER (bit error rate) less than 10^{-12} [6], [8], otherwise performance of the networks degrades significantly [9].

There are also emerging applications that require latency and jitter bounds in larger scale networks. Machine-to-machine communications for 'cloudified' industrial and robotic automation involves moderate-to-large scale networks. This type of communications requires very fine-grained timing accuracy for the dispersion of control commands and for the collection of telemetry data over a wide area [10]. ITU-T SG-13 has defined such services to support critical grade reliability and extremely low as well as highly precise latency for the delivery of packets [10]. This is because some industrial controllers require very precise synchronization and spacing of telemetry streams and control data, facilitating, for example, precise operation of robotic effectors along multiple degrees-of-freedom [10]. Despite the fact that even loose jitter bound requirements are hard to be satisfied in large scale networks, the importance of 'on-time service' as well as 'in-time service' is emphasized in [10].

For another example, the 3GPP TS23.501 specifies the periodic automation control flows with data rates of only 0.1∼0.2Mbps to have the upper bound 10ms [11]. See Table 2. ITS stands for intelligent transport systems, in the table.

TABLE 2. Flow types characteristics specified in 3GPP TS23.501.

5QI	Data rate (Mbps)	Burst (Max packet size, bits)	Max Latency (ms)	Example Service
82	0.1	2040	10	Automation
83	0.2	10832	30	Automation
84	0.3	10832	30	ITS
85	0.3	2040	5	Electricity Distribution

B. PROBLEM STATEMENT

We consider the problem of guaranteeing the jitter bound in arbitrary sized networks with any type of topology, with random dynamic input traffic. The jitter is defined to be the latency difference of two packets within a flow, not a difference from a clock signal or from an average latency, as it is clearly summarized in RFC 3393 [12].

In large scale networks, the end-nodes join/leave, and the flows are dynamically generated and terminated. Achieving satisfactory deterministic performance in such an environment would be challenging.

C. PREVIOUS WORKS ON JITTER MINIMIZATION

For small networks such as in-car networks or 5G fronthaul networks, IEEE TSN task group defines a set of solutions of latency and jitter minimization [13]. The solutions rely on the time-synchronization of every node in the network and slot scheduling that is coordinated among nodes throughout a network. Note that we use the term time-synchronization, instead of clock-synchronization. The time or phase-synchronized systems are defined such that significant events occur at the same time with the same rate [14]. In contrast, in the frequency-synchronized systems the events occur at the same rate but not necessarily at the same time.

The synchronized nodes allocate service time slots to a set of predefined traffic flows such that the flows are continuously served by consecutive nodes. This is called 'peristalsis', and is the heart of the TSN solutions. The complexity resides at the allocation of slots to a set of flows, however, such that the performance requirements of various flows are met in given networks. This slot allocation task is called 'slot scheduling' or just 'scheduling'. It is not to be confused with the traditional packet scheduling, with which it is decided when the packets in the queues are served.

The slot scheduling is a time and resource consuming task. When the network is large, it is quite a burden to the network operator. A conflicting schedule is defined as one that schedules more than one of the assigned slots at the same time. Finding a non-conflicting schedule of packets is proven to be NP-complete, even with fixed sized slots similar to a TDMA [7]. [15] and [16] proved that the problem of producing a non-conflicting schedule to multiplex multiple flows can be reduced to the classical graph-coloring problem, which is known to be NP-complete. Even the heuristic algorithms take long for the slot scheduling as the number of flows increase. With randomized topologies with a set of time-sensitive flows randomly generated, the time taken for slot scheduling is 10^2 seconds with 200 flows [17]. It grows to almost 10^4 seconds when the number of flows is 1000 [17]. For dynamic environments where critical flows appear and disappear over time, runtime reconfigurations are necessary [18], which makes the scheduling problem even harder. Moreover, in cases the propagation delay is significant compared to the slot length, synchronization and coordination among the nodes may become ineffective, therefore a more careful slot allocation is necessary. An on-line time-slot reconfiguration framework for TSN based on software defined networking (SDN) context is also proposed [19]. The joint optimization of admission control, slot scheduling, and routing problems in TSN is considered in [19].

The time-synchronization requirement across every node in the network has two difficulties. First, time synchronization function implementation may impose hardware support, thus too much overhead to lightweight embedded nodes. Second, the synchronization accuracy may not be up to the level of traffic requirements. There are two representative

protocols for network-wide time-synchronization, network time protocol (NTP) and precision time protocol (PTP) [20], [21]. NTP uses timestamps in the application layer and has a limited synchronization accuracy to 1ms [22]. In software based implementations, PTP achieves mean synchronization accuracy of 30 μs and standard deviation of 54 μs [23]. Those results can be slightly improved by using real-time operating systems and communication channels, but even in this case reaching sub-μs synchronization accuracy remains a tough task [24]. PTP can reach the accuracy to 1μs or lower, provided with a hardware timestamp function and compensation for transmission path delay, for a node 30 hops away from the grandmaster [25]. Considering the CPRI's jitter requirements of 65$\sim$100ns [6], the hardware implementation is a must for such an application. However, this comes at the cost of being more expensive both in terms of components and development costs.

PTP measures the one-way transit delay from a master to a slave by exchanging the messages in both directions with time-stamps. Based on the obtained latency value and the time-stamp of the master, the slave node calculates the clock offset and corrects it. For this message exchanging protocol to work accurately, a few assumptions must be made. 1) First, this exchange of messages happens over a time interval so small that this offset can safely be considered constant over that interval. 2) Second, the transit time of a message going from the master to a slave is equal to the transit time of a message going from the slave to the master. 3) Third, both the master and slave can accurately measure the instance they send or receive a message. The degree to which these assumptions hold true determines the accuracy of the clock at the slave device [21]. It is easy to see that these assumptions, especially the second one, may go wrong in networks with a large number of hops.

With all the efforts mentioned above, however, there is no system that can guarantee a jitter bound in general networks with arbitrary traffic yet. For a limited topology, such as a single-hop CPRI network, there are researches about achieving zero jitter with packet-level slot scheduling methods in lightly utilized environments [7]. This work shows that even for a smallest network with lightly loaded traffic, it is necessary to micro-manage the slots with the size of a single packet transmission, and to reorder slots dynamically, in order for achieving zero jitter [7].

Another work on jitter bound [26], [27] utilized an input-buffered switch architecture with a slightly different type of slot scheduling. An IP packet is divided into 64byte 'cells' and the scheduler allocates fixed slots to cells from input ports at every node. A frame is composed of a finite number of slots (F). This work requires matrix operations of complexity $\max(\mathrm{O}(N^2), \mathrm{O}(NF))$, where N is the number of ports in a switch. It tried to minimize jitter of video traffic at every node, and the extra jitter is removed from the 'playback buffer' at the end of the network, based on the prior knowledge of a fixed inter-arrival times (i.e. 33ms) of the video frames. No time-synchronizations or time-stamps are necessary [28].

In real time protocol (RTP) and RTP control protocol (RTCP), the timestamps are used mainly for two purposes. The first is to measure the time related performances and to feedback for the adaptive encoding or the faults diagnosis [29]. The second is to control the jitter at the application playback buffer. Additionally, since the time-stamps of RTP/RTCP may have the same format with NTP, it can be used for network time-synchronization. The time-stamps in RTP, however, is to be set to represent the sampling instance of the periodic multimedia data, or the intended presentation instance of the data, not the actual transmission instance of a packet. For example, if a video frame consists of multiple packets, these packets should have the same time-stamp value. This is because the RTP is usually above UDP, a transport layer protocol, thus handled at the OS level. It is hard to predict precisely the transmission instance before actual transmission. The time-stamps in RTP for the jitter control, therefore, does not play a critical role if the receiver is already aware of the flow's presentation period, as in the case of [28].

D. CONTRIBUTIONS AND CONTENTS

This paper proposes a new framework for jitter bound guarantee. The framework does not require a time-synchronization, or a slotted operation. It requires three components; 1) a network that can guarantee latency upper bounds, 2) packets with relative time-stamps, and 3) buffers that are able to hold packets for pre-defined intervals. The relative time-stamps is the time-stamps inscribed by traffic sources that are not synchronized with other nodes. The key idea is that, based on the latency bound value provided by the network, the first packet of a flow is buffered long enough to make sure that the inter-departure times of all the subsequent packets of the flow are similar to their inter-arrival times. While the time-stamps are usually used for latency/jitter measurement [12], [29] or clock synchronization [20], [21], in this work we use it as the primary tool for jitter bound guarantee. It is proven that this framework can guarantee a jitter upper bound of any desired amount, even zero jitter. It is also proven that the framework still guarantees latency upper bounds as well.

We examine the network solutions, which are able to guarantee latency bounds, to include into the proposed framework. We then analyze and compare the latency and jitter performances of the proposed framework with the TSN framework. The insight we gain through the analysis is twofold. First, the isolation of critical packets or flows from the bursts of other traffic is essential for latency and jitter minimization. The work-conserving fair schedulers are shown to be good at the isolation of the flows. Second, the non-work conserving nature of the slotted operation is disadvantageous to both the latency and the jitter performances. Such analytical results are confirmed with the simulations.

In Section II, we describe the proposed framework and explain its properties. It is shown that even zero jitter can be guaranteed with the framework. In Section III, a brief background of various networks for latency guarantee

is given. In Section IV, the detailed descriptions on the models for networks mentioned in the previous section are given. In Section V, an analysis of an example network with light load is given for a case study. The performances of existing solutions for latency guarantee are compared based on the simple exemplary network. Based on the latency upper bound values, we suggest the jitter bound values achievable with each solution. We validate the analysis with simulations in Section VI. The conclusions and the future works are given in Section VII and VIII, respectively.

II. PROPOSED FRAMEWORK FOR JITTER BOUND

In this section we describe a novel framework for jitter upper bound guarantee. The framework requires a network that guarantees latency upper bounds; packets with relative time-stamps inscribed with the clock of the source, or of the network ingress interface, not necessarily synchronized with the other nodes; and a buffer before a destination, which can hold the packets destined for the destination for a predetermined interval. Figure 1 depicts the overall architecture of the proposed framework. Only a single flow is depicted between the source and the destination in Figure 1. The arrival (a_n), departure (b_n), and buffer-out (c_n) instances of n^{th} packet of a flow are depicted. The E2E latency and the 'E2E buffered latency' are defined to be (b_n-a_n) and (c_n-a_n), respectively.

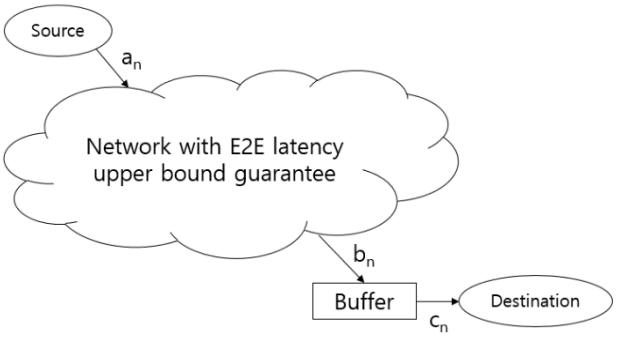


FIGURE 1. Architecture of the proposed framework for jitter bound. Packets' arrival (a_n), departure (b_n), and buffer-out (c_n) instances of n^{th} packet of a flow are depicted. E2E latency (b_n-a_n) and E2E buffered latency (c_n-a_n) should not be confused with each other.

The buffer is assumed to support as many as the number of the flows destined for the destination. In cases where the buffer is not suitable to be placed within an end station, the network can attach the buffering function at the boundary. The destination in Figure 1 can also be a small deterministic network, like a TSN synchronous network.

We assume there is a mechanism for time-stamping the arrival instances to the packets. One may use the time-stamping function in the RTP over UDP, or TCP. Either the source or the network ingress interface may stamp the packet. In the case that the source stamps, the time-stamp value is the packet departure instance from the source, which is only a propagation time away from the packet arrival instance to the network. Note that the source and the destination need not to share the synchronized clock. All we need to know in the proposed framework is the differences between the time-stamps, i.e. the information about the relative arrival instances.

We also assume the latency upper bound of the flow is guaranteed by the network. Let's denote it with U. We also assume that a latency lower bound is provided by the network to the flow, which is denoted by W. The lower bound W may be contributed from the transmission and propagation delays within the network. The buffer holds packets in a flow according to predefined intervals. The decision of the buffering intervals involves the time-stamp within each packet as the following.

Let the arrival instance of n_{th} packet of a flow be a_n. Similarly let b_n be the departure time from the network, of n_{th} packet. Then a_1 and b_1 are the arrival and departure instances of the first packet of the flow, respectively. The first packet of a flow is defined to be the first packet generated by the source, among all the packets that belong to the flow. Let's say m is some value between W and U, $W \leq m \leq U$. Let us summarize the symbols used in this section in Table 3.

TABLE 3. Mathematical symbols for the framework description.

Symbol	Quantity
a_n	Arrival instance of the n_{th} packet of the flow to the network
b_n	Departure instance of the n_{th} packet of the flow from the network
c_n	Buffer-out instance of the n_{th} packet of the flow
$b_n - a_n$	E2E latency of the n_{th} packet
$c_n - a_n$	E2E buffered latency of the n_{th} packet
U	E2E latency upper bound of the flow guaranteed by the network
W	E2E latency lower bound of the flow guaranteed by the network
m	Jitter control parameter of the framework, $W \leq m \leq U$.

The basic rule for the buffer holding interval decision is given as follows.

Rule for the buffer holding interval:

- The buffer holds the first packet with the interval $(m - W)$, for some m, $W \leq m \leq U$. The buffer-out instance of the first packet c_1 is then $(b_1 + m - W)$.
- The buffer holds n_{th} packet until the instance $\max\{b_n, c_1 + (a_n - a_1)\}$, for any $n > 1$. ■

The second rule implies that a packet should be held in the buffer to make its inter-buffer-out time ($c_n - c_1$) equal to the inter-arrival time ($a_n - a_1$). However, when its departure from the network is too late, thus the inter-buffer-out time should be larger than the inter-arrival time, then just pass the buffer ($c_n = b_n$).

These two rules can be elaborated as the following.

Feasibility of the Algorithm 1: The buffer requires information on W, U, b_1c_1, b_n, and $(a_n - a_1)$. We assume that W, U are informed by the network. The knowledge about b_1 and b_n are easily obtained by the buffer with its own clock.

Algorithm 1 Buffer Holding Interval Decision

01: **procedure** BUFFER(m, PKT)
▷ m is a preset value based on W, U, and the jitter bound ▷ $W \le m \le U$
▷PKT is a packet just received by the buffer
02: **if** first packet in the flow **then**
03: hold the packet with the interval (m-W)
▷The buffer-out instance of the first packet c_1 is then $(b_1 + m - W)$
04: TRANSMIT the packet at the decided instance
05: **while** there is a packet already arrived before the first packet
06: hold the packet until the instance $\max\{b_n, c_1 + (a_n - a_1)\}$ ▷Let's say this is n^{th} packet, n>1
07: TRANSMIT the packet at the decided instance
08: **end while**
09: **else if** the first packet has already arrived **then**
10: hold n_{th} packet until the instance $\max\{b_n, c_1 + (a_n - a_1)\}$ ▷Let's say this is n^{th} packet, n>1
11: TRANSMIT the packet at the decided instance
12: **else** wait for another packet arrival
13: **end if**

The buffer has to keep the record of the time instance c_1. Finally the time difference $(a_n - a_1)$ can be calculated from the difference of the time-stamps of the packets, which have been written at the source. Note that the source clock does not have to be synchronized with the buffer clock, i.e. the buffer does not need to know the exact values of a_n or a_1.

The implementation of the line 6 and 10 in Algorithm 1 is feasible since $\max\{b_n, c_1 + (a_n - a_1)\}$ is greater than or equal to b_n, the packet departure instance of the n_{th} packet from the network, by definition.

The buffer has to be able to identify the first packet of a flow, in order to identify the instance b_1 and the relative time-stamp values representing a_1. If a FIFO property is guaranteed in the network, then this is trivial. Otherwise, this is achievable by a flag at the header, indicating the packet is indeed the first packet. A sequence number written in the packet header, such as the one in RTP, would work as well. In networks without the FIFO property guarantee, if some of the earlier packets (e.g. 2^{nd} or 3^{rd} packets of a flow) arrive to the buffer sooner than the first packet, than they will be buffered until the first packet's buffer-out plus the additional interval, as specified in the algorithm. ■

Figure 2 depicts the relationships between the arrival, departure, and buffer-out instances of packets of the flow under observation.

The following theorems hold.

Theorem 1: (Upper bound of the E2E buffered latency). The latency from the packet arrival to the buffer-out instances, $(c_n - a_n)$, is upper bound by $(m + U - W)$.

Proof: By definition,

$$c_n - a_n = \max\{b_n, c_1 + (a_n - a_1)\} - a_n$$

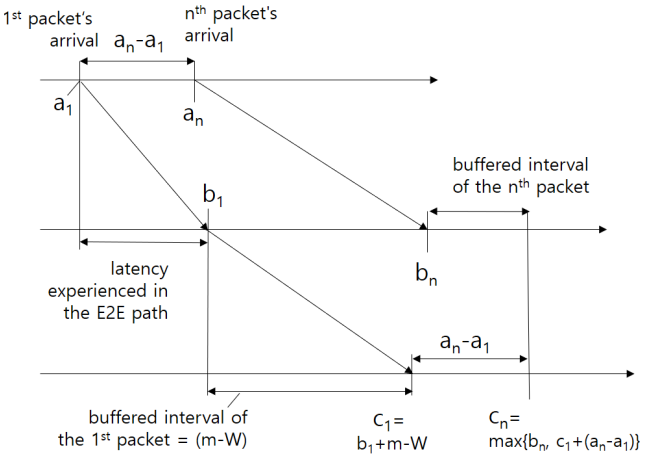


FIGURE 2. Relationship of packets' arrival (a_n), departure (b_n), and buffer-out (c_n) instances.

$$= \max\{b_n - a_n, b_1 + m - W - a_1\}$$
$$= \max\{b_n - a_n, m - W + (b_1 - a_1)\}$$
$$\le \max\{U, m - W + U\} = m - W + U,$$

since $b_n - a_n \le U$ for any n, and $m \ge W$. The theorem follows. ■

Theorem 2: (Lower bound of the E2E buffered latency). The latency from the packet arrival to the buffer-out instances, $(c_n - a_n)$, is lower bounded by m.

Proof: By definition,

$$c_n - a_n = \max\{b_n, c_1 + (a_n - a_1)\} - a_n$$
$$= \max\{b_n - a_n, m - W + (b_1 - a_1)\}$$
$$\ge \max\{W, m - W + W\} = m,$$

since $b_n - a_n \ge W$ for any n, and $m \ge W$. The theorem follows. ■

Now let us define the jitter, or the latency difference between a pair of packets, as follows.

Definition : (Jitter). The jitter between the i_{th} packet and j_{th} packet of a flow is defined to be $r_{ij} = |(c_i - a_i) - (c_j - a_j)|$.

Theorem 3: (Upper bound of the jitter). The jitter is upper bounded by (U-m).

Proof: Let's define $r_n = r_{n1} = (c_n - a_n) - (c_1 - a_1)$. Remember that $c_1 = (b_1 + m - W)$ and $c_n = \max\{b_n, c_1 + (a_n - a_1)\}$. From the definition,

$$r_n = (c_n - c_1) - (a_n - a_1)$$
$$= \max\{b_n - c_1, a_n - a_1\} - (a_n - a_1)$$
$$= \max\{b_n - c_1 - (a_n - a_1), 0\}$$
$$= \max\{b_n - b_1 - m + W - (a_n - a_1), 0\}$$
$$= \max\{(b_n - a_n) - m + W - (b_1 - a_1), 0\}$$
$$\le \max\{U - m + W - W, 0\} = U - m, \text{ since } U \ge m.$$

The jitter between packets i and j, r_{ij}, can be rewritten such as $r_{ij} = |(c_i - c_1) - (a_i - a_1) - (c_j - c_1) + (a_j - a_1)| = |r_i - r_j|$. Since $0 \le r_i \le U - m$ and $0 \le r_j \le U - m$, the jitter $r_{ij} \le U - m$, for any i, $j > 0$. The theorem follows. ■

Example : Now, let's say we have a flow requesting the E2E buffered latency bound of 10ms, and jitter bound 1ms. Then from Theorem 1, (m+U−W) = 10 ms, and from Theorem 3, (U − m) = 1 ms. Based on these equations we obtain U = 5.5ms+W/2 and m = 4.5ms+W/2. As such, during the call setup process, upon the flow's requested specifications, the network and the buffer may assign $U = (5.5\text{ms} + \text{W}/2)$ of the actual E2E network latency upper bound, and $m = (4.5\text{ms}+\text{W}/2)$ parameter for the buffering. ■

As an extreme case, if one wants to achieve an absolute synchronization, i.e. the inter-departure times of the output packets $(c_n - c_{n-1})$ are exactly the same with the inter-arrival times $(a_n - a_{n-1})$, then one may set the jitter to be equal to zero. In this case we can achieve this synchronization by setting m = U, i.e. by holding the first packet long enough that the E2E buffered latency certainly exceeds the U. Then the buffered latency upper bound becomes 2U-W, which is close to 2U when W is negligible. Note that letting m = U does not always mean the E2E buffered latency becomes 2U. It is just the upper bound in the worst case in which the first packet had already experienced the latency U in the network.

III. NETWORK SOLUTIONS FOR LATENCY GUARANTEE

In this section we review the works related to the latency guarantee or, more generally, quality of service (QoS) control mechanisms researched over three decades. We also examine the possible candidates for adopting as the solution for latency guarantee network necessary in our framework.

The flow-based approaches such as the integrated services (IntServ) framework, or more specifically its Guaranteed Service framework, has been known to provide the latency bound guarantee [30], [31]. However, the IntServ's packet scheduling complexity is proportional to the number of flows, which grows to millions in core networks. This scheduling complexity prohibits the IntServ from being implemented in real networks.

The differentiated services (DiffServ) framework provides relative performance differentiation with 8 or 32 queues for each priority class [32]. Flows belonging to a class are put into a queue. The queues are served with strict priority. Because of such simplicity, The DiffServ has been adopted in the current Internet. However, the maximum burst of a flow increases according to the sum of all the flows' max bursts within a queue. When there is a cycle in a network, the max burst grows to infinity, so does the latency bound. Therefore, the DiffServ framework does not provide a latency bound in a general topology network. Recently it has proven that a network that can be modelled as a multi-stage switch without a cycle can guarantee latency bounds, if work-conserving schedulers are used and input traffic is properly regulated [33].

There are efforts to suppress the burst accumulation caused by the flow aggregation, with interim regulators [34], [35]. [34] pioneered the flow regulation in a node to suppress the burst accumulation. [35] has devised a fair scheduler that works as a regulator at the same time. IEEE 802.1 TSN also has standardized multiple shapers, or synonymously regulators, although their roles and aims are slightly different from each other. The TSN standard aims to guarantee latency upper bound in single domain networks, therefore in relatively small scale networks. The P802.1Qcr asynchronous traffic shaping (ATS) technique [36] presented in TSN employs an output port, with interleaved regulators (IR) per input port and a strict priority class-based FIFO queuing system side by side. IR is a single queue system that examines the packet at the head of the queue (HOQ), and lets it leave as soon as it is qualified according to the regulation rule of the flow the packet belongs to. The rest of the packets in the queue are held until the HOQ packet leaves, even if they are already qualified. However, it is proven that a minimal IR does not increase the delay upper bound of the attached FIFO system, such as the class-based FIFO queue employed in TSN [37]. The ATS framework requires an IR per input port at every output port of every node. The regulation at every node, because of its non-work conserving characteristic, degrades the probabilistic performance, such as the average delay. It also implies increased implementation cost.

The IntServ, the DiffServ, and the TSN ATS deal with the latency upper bound, but they do not provide the jitter bound guarantee. TSN task group has standardized multiple functional components for jitter-sensitive services. Among them, the 802.1Qbv time sensitive queues (also known as the Time Aware Shaper, TAS), and 802.1Qch cyclic queuing and forwarding (CQF) are built for jitter minimization as well as latency guarantee.

The TAS defines the ''gate'' to each queue. The gates are either open or closed on a time slot. Multiple gates may open simultaneously. In such a case the strict priority scheduling applies to the open queues. A central network controller (CNC) determines which gates to open and when to open. CQF function coordinates the slot openings in adjacent nodes, such that flows traversing the nodes experience the minimum latency. Based on these functions, TSN supports a model for deterministic packet services. In this work we will call these functions collectively as the TSN synchronous approach. The TSN synchronous approach requires three major efforts; 1) the time synchronization along the nodes in the network, 2) the slot scheduling and coordination by a central entity, and 3) feasibility test for flow admission control. These requirements significantly impact the scalability of a network. The detailed operation of the TSN synchronous approach is described in Section IV.A.

Recently it was proposed and standardized, in ITU-T SG13, a hybrid approach between the IntServ and the TSN ATS [38], [39]. It aims for the latency bound guarantee in large-scale networks. Within an aggregation domain the flows are aggregated into a flow aggregate (FA) according to their input/output port pair, or additionally according to flows' requirements and characteristics. The FAs are treated with separated queues and fair-queuing schedulers, thus maintaining the FIFO property throughout the

aggregation domain. Then at the boundary of the aggregation domain the minimal IRs per FA are placed for the burst suppression. See Figure 3. We call this approach the FAIR (flow aggregate & interleaved regulator) framework. The FAIR is proven to guarantee latency upper bound with modest complexity in large scale networks. It was shown that the FAIR even performs better than both the IntServ and the ATS, in symmetric network topologies [38]. The detailed operation of the FAIR is described in Section IV.C.

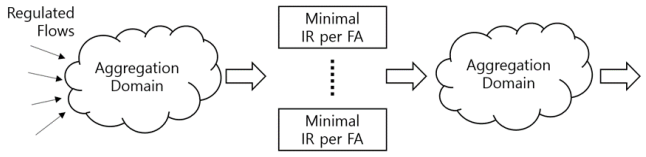


FIGURE 3. The Flow aggregate-interleaved regulator (FAIR) framework [38].

IV. MODELING OF THE SOLUTIONS FOR LATENCY GUARANTEE

In this section we review the detailed features of three different solutions for the latency guarantee network, namely the TSN synchronous approach, the DiffServ, and the FAIR framework. Then the three solutions are modeled to be used in the case study of Section V, where the latency upper bounds of the three solutions are analyzed. While the DiffServ and the FAIR are relatively simple to understand, the TSN has a lot of features and parameters to configure. The parameters related to the slot and cycle operations are carefully examined. The purpose of this section is twofold. First, it is to examine the feasibility of adopting the solutions as a component of the proposed framework. Second, it is to develop the accurate simulations setup. The simulation results are described in Section V.

A. THE TSN SYNCHRONOUS APPROACH

A TSN node's output port is essentially a combination of 1) flow reservation and regulation functions, 2) IR per input ports, 3) class based queues, 4) credit-based shapers which is a combination of a starvation-prevention function and a burst accumulation prevention function, for higher priority classes queues, 5) gates that selectively open/close the queues based on time slot, i.e. 802.1Qbv time-aware shaper (TAS), and 6) strict priority scheduler, and finally 7) frame preemption for express class traffic. See Figure 4.

Additionally, 8) the selective openings of the queues are synchronized and coordinated among successive nodes, i.e. 802.1Qch cyclic queuing and forwarding, for the best performance of the network. The coordinated openings of the queues, or the coordinated 'slot scheduling', are controlled by a central network controller (CNC) in the network.

These components can be selectively combined and implemented, according to service specific requirements. For example, components ② ③⑥ only may be combined together and can guarantee latency upper bound in any topology network. We will denote the TSN ATS as such a combination.

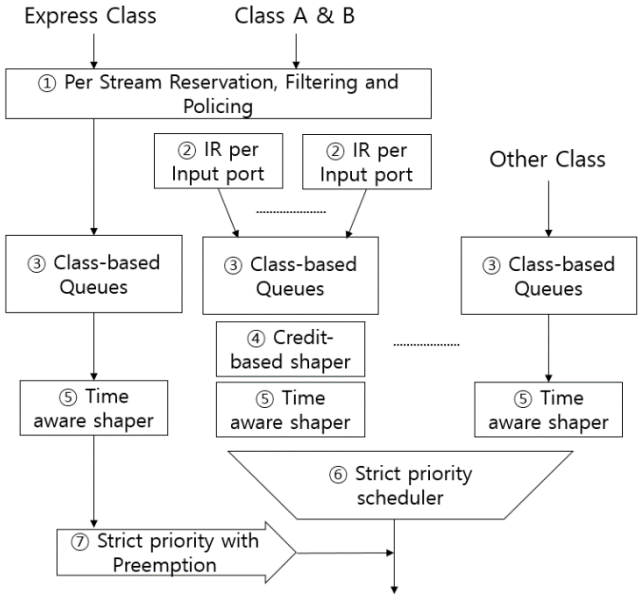
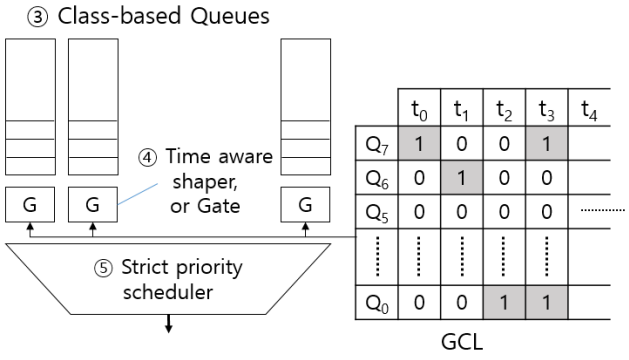


FIGURE 4. Illustration of an output port of a TSN node and its functional entities.

Another example combination may include ③⑥ only. This combination represents essentially the DiffServ framework, and can guarantee latency upper bound in network topologies without cycles, such as tree networks. We will denote this combination as the DiffServ. Yet another example, including ③⑤⑥⑧ only, is a coordinated gate opening scheme, which is the core of the TSN standard, and can be called the 'TSN synchronous approach' collectively.

Basically there is no limitation on the slot start time and the length, in general slot scheduling problems. Only the implementation complexity prohibits the degree of flexibility of slot generation. Let us examine how TSN suggests in this problem. Figure 5 depicts the simplified logic of the output port specified in the 802.1Qbv TAS function. Each switch output port has 8 priority queues, and the arriving flow enters different queues according to the priority or the service policy. All the slots have a fixed length. The slots for the queues are synchronized, i.e. the slot intervals exactly overlap. Each queue has a gate (G in Figure 5). Gate control list (GCL)



	t_0	t_1	t_2	t_3	t_4
Q_7	1	0	0	1	
Q_6	0	1	0	0	
Q_5	0	0	0	0	
⋮	⋮	⋮	⋮	⋮	⋮
Q_0	0	0	1	1	

FIGURE 5. The Traffic aware shaper (TAS) with gate control list (GCL).

governs the gates' open or closing at every slot. With the value 1, we represent the gate of the queue is open, and the flow in the corresponding queue can be transmitted; and with 0 we mean it is closed. Note that more than one gate can be open in a slot. Gate control list (GCL) is a finite list, and this list repeats with a cycle, so the gate will be open/closed periodically according to the cycle.

Usually in the 802.1Qbv TAS, GCL is cycled according to a 'hyper period' based on the least common multiple of all the periods of the flows, regardless of the queues they belong to. For example, if we have three flows with period 3, 4, and 6, in any number of queues, then the hyper period has twelve slots. In this case, once in every twelve slots all three flows' gates can be open simultaneously. The size of the hyper period thus GCL is usually very large when there are many flows, which increases the complexity of slot scheduling [40].

Note that based on the GCL, the slot length in 802.1Qbv for all the queues are the same fixed value. How long the slot should be is an open question in each implementation in different circumstances. It should be set at a balanced point in between the complexity and the precision. At an extreme it can be as short as a byte. Consecutive slots, as many as the bytes count of a packet, must be open for a packet to be serviced. One may see those consecutive slots a single 'variable length' slot. In such a case the cycle period would be too long, and the scheduling too complex. Maybe a different method for the slot assignment description than the GLC should be used in this extreme case. At another extreme the slot can be as long as to accommodate all the maximum bursts from all the input ports. This may be a feasible implementation and the most deterministic in its operation. However, a long slot inevitably yields a long interval between the open slots, which results in increased latency when the input bursts are dispersed evenly over time. It is a well-known drawback of the TAS that unsynchronized input flows have to wait for the next slot opening. It was also shown, through a hardware-based experiment with a star-topology 3-hop TSN network with the TAS and the CQF, that the average latency and the jitter increase as the slot length increases [41].

Once the length of the slot is decided, then the next question is when to open the gates. At one extreme, if all the queues are always open, then the whole node works as a strict priority scheduler. At the other extreme, if the queues are strictly open one at a time, in a round-robin fashion, then the whole node works as a time-division multiplexing (TDM) scheduler. In some sense, the TSN network can be considered as a set of TDM nodes with strict priority scheduler, with a flexible slot-scheduling policy specified by a central controller.

For a flow, if the cascaded nodes are successfully coordinated, the queue assigned to the flow may open as soon as its packet arrives at a node, and this may happen at every node for every packet in the flow. This will be the best case for a flow, and the intended operation of TSN 802.1Qch CQF. This optimal scheduling is feasible in small networks with low utilization with static flows and topology. On the other hand, the queue for a flow may be closed just before the arrival of the packet of the flow, at every node for every packet. This will be the worst case. It is easy to see that the slot scheduling plays the key role in the TSN network performance.

The slot scheduling techniques usually focus on, however, the optimization of the flow acceptance ratio therefore the overall revenue of the network. The optimization of a specific flow's performance is not the primary objective. As such, we can expect a treatment to flow somewhere in between the best case and the worst case described above. We model the TSN synchronous approach to work as follows. First, the lengths of the slots are different at each node and is set according to the maximum input burst into the node. Second, the nodes are synchronized and coordinated, such that immediately after the slot completion at an upstream node the adjacent downstream node's slot starts.

B. THE DIFFSERV

The DiffServ framework can be modelled with class-based queues and the strict priority scheduler. With the TSN's context, the DiffServ is composed of the components ② and ⑤ in Figure 4. Additionally, we assume for simplicity that every lower priority traffic can be preempted with any of higher priority packets.

C. THE FAIR FRAMEWORK

In the FAIR framework [38], the whole network is divided into a few aggregation domains (ADs), and the flows that share the same {input, output port} pair to/from an AD are aggregated into a flow aggregate (FA). The FIFO property of the FA within the AD has to be preserved. Minimal IRs per FA are implemented at the boundary of the AD.

The minimal IRs at any 'system' boundaries suppress the burst accumulations with the latency upper bounds intact [37], given the conditions below are satisfied:

(1) Every flow into a 'system' conforms to an arrival curve with parameters {average arrival rate, maximum burst size}.

(2) The system preserves the packets' FIFO property.

(3) The IR regulates every flow to reproduce the arrival characteristics at the ingress of the system.

(4) (Minimal IR) IR transmits immediately when the packet at the head of the queue meets the output condition. Such IR is called a minimal IR.

(5) IR provides zero latency, including transmission latency, for packets satisfying output conditions.

Note that the condition (5) is inferred from Theorem 3 in [37]. It is a necessary condition for (4) as well. Theorem 3 of [37] states that the latency bound of a FIFO system is not increased by the attached minimal IR. In order for Theorem 3 holds for the packet that experiences the maximum latency in the FIFO system, the minimal IR has to provide the zero latency. A look-ahead function and a cut-through capability are necessary for actual implementation of a minimal IR.

In the TSN ATS, the 'FIFO system' in which these conditions are met is an in/out ports pair of a switching node.

In contrast, the FAIR framework extends the FIFO system to an AD. The FAIR framework works as the followings, based on the conditions described above:

- Flows are categorized into high priority and low priority.
- Low priority flows are put in a single FIFO queue at the output port of all the nodes and scheduled in strict priority mode with, preferably, preemption.
- High priority flows are handled as follows.
  - The network is divided into several ADs.
  - Within an AD, the flows with the same {input, output port} to/from the AD are aggregated into a single FA.
  - In a node, a fair queuing-based scheduling is performed per FA with a queue for each FA. This operation lets the AD, for an FA, become a FIFO system in Condition (2) above.
  - Minimal IRs per FA are placed at the boundaries of ADs. See Figure 3 in Section III.
  - Only the flows conforming the initial arrival characteristics or the flows from the IR are accepted to an AD.

For a small network, the whole network can be a single AD, thus the IRs may be omitted, as in the case study of the following section.

V. CASE STUDY

We consider a simple network with a set of small number of time sensitive flows, which is similar to the one originally analyzed in [42], [43]. It is also similar in terms of structure to the prototype Ethernet network developed by an automotive OEM [44].

The traffic is made up of three classes whose characteristics are summarized in Table 4. ADAS stands for advanced driver assistant system, in the table. Characteristics of the flows and their proportions are the same as in [42], [45] and inspired from the case-studies provided by [46], [47].

TABLE 4. Characteristics of the three types of traffic used in [46], [47].

Traffic type	Characteristics
Audio flows	128 or 256byte payload 1.25ms period 5 or 10ms deadline 7/46 proportion
Video flows	ADAS or Vision 30*1500byte or 15*1500byte frame 33ms period (30 frame per sec) 10 or 30ms deadline 7/46 proportion
Command & Control flows	53~300byte payload 5 ~ 80ms period deadline is equal to the period 32/46 proportion

The video flow emits a larger burst compared to the other types of flows. The number of C&C flows are more than double the number of the other two types of flows combined.

We have further simplified the flows characteristics such that the Audio flows (A flows) emit 256byte packet every 1.25ms, Video flows (V flows) emit 30∗1500byte bursts with 33ms period, and C&C flows (C flows) emit 300byte packet every 5ms. Four A flows and a V flow (red curve) share the route, so do 20 C&C flows, as depicted in Figure 6.

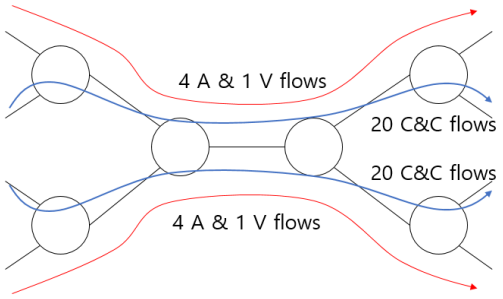


FIGURE 6. Network topology and the flows in the case study.

Link Capacity for all the links are set to be 1Gbps, which is common with Gigabit Ethernet nowadays. With the notation for a flow {Packet length, Max burst, Arrival rate}, Audio flow's parameter set is {256B, 256B, 256B/1.25ms} ∼ {2Kbit, 2K, 1.6Mbps}, Video flows' parameter set is {1500B, 30∗1500B, 30∗1500B/33ms} ∼ {12Kbit, 360Kbit, 11Mbps}, and C&C flows' parameter set is {300B, 300B, 300B/5ms} ∼ {2400bit, 2400, 480Kbps}. It is summarized in Table 5. In the studied case, the total arrival rate is well below the link capacity. It is 5.4% of the link capacity. It is not uncommon that the high priority traffic occupies only a small fraction of the link. The case study with high utilization is partly covered in Section V.E.

TABLE 5. Traffic types and their approximated flow parameters used in the case study.

Symbol	Traffic type	{Packet length, Maximum burst size, Arrival rate} of a flow
A	Audio	{2Kbit, 2Kbit, 1.6Mbps}
V	Video	{12Kbit, 360Kbit, 11Mbps}
C	Command & Control	{2.4Kbit, 2.4Kbit, 480Kbps}

The packet length, maximum burst size, and the period of a flow are fixed. The only randomness that can be introduced in the analysis is the start time of the period, or the phase. The input traffic phase usually cannot be controlled in general. Since we are interested in the worst case scenario, however, we only consider the input pattern that produces the worst performance of the network. We consider the case that all the flows' periods (or the burst arrivals) start exactly at the same time. The packet under observation arrives last among the burst.

Given the topology, the input flows, and their destination; we will consider three types of solutions that all guarantee latency bound. They are the TSN synchronous approach, the DiffServ, and the FAIR framework. The three types of

traffic are assumed to have the same high priority. We will observe each flow and their latency upper bounds. We will show that the DiffServ and the FAIR framework, with the buffers at the boundary, work as well as or even better than the sophisticated TSN synchronous approach.

A. LATENCY BOUND WITH THE TSN SYNCHRONOUS APPROACH

We follow the model in Section IV.A. We assume that the propagation delay at every link is negligible. We assume for simplicity, that any lower priority traffic is preempted by the higher priority traffic. In order to minimize the latency, we also assume that the slots are of variable length. The slots for high priority traffic have to open according to the input burst amount of the high priority traffic. The slots in cascading nodes are synchronized and coordinated such that right after a node's slot completion, the downstream node's slot is open. The slots in neighbor nodes do not overlap so that every packet in an upstream node can be served in the downstream node's slot right after its service completion in the current node.

The detailed slot assignments at each node are depicted in Figure 7. The inter-arrival intervals for A, V, and C flows are 1.25ms, 33ms, and 5ms, respectively. The hyper-period of the slot assignment is therefore 165ms, based on the least common multiple of these three intervals. Note that in Figure 7, the slots are used by either A only, A&C only, and A, C, & V together, in the first three nodes. For example, the A flows collide with the C flows every four slots, and share the slot. Also note that there are slots used for the other combinations of flows, which are not depicted in Figure 7.

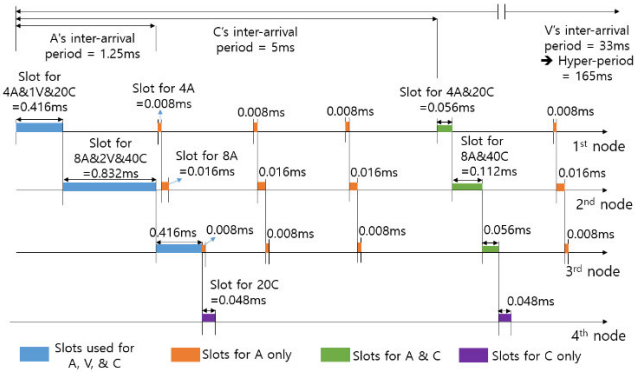


FIGURE 7. Periodic slot assignments for each node in the TSN synchronous approach, with the hyper-period of 165ms.

The precise variable length slots used in our model may not be easily attained by the current TAS implementations with the GCL that is a simple record of 0/1 on fixed length slots. A coarse fixed length slot assignment imposes a loose latency bound. The slot assignment policy in our model can be seen as the optimum yet complex implementation of TAS.

The worst case packet latency for C&C flows is analyzed in Figure 8. At the 1st node, the slot is long enough to just accommodate the maximum burst. The worst packet's arrival is a little behind all the other bursts. (We mean by

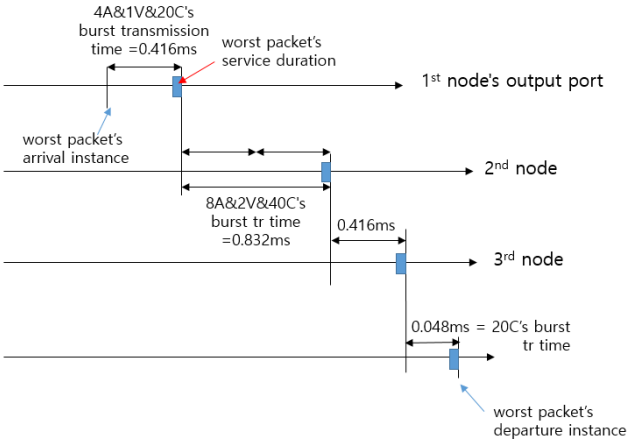


FIGURE 8. The worst case scenario of C&C flows in the TSN synchronous approach with the variable length slot scheduling. The latency is 1.712ms in this scenario.

the 'worst packet' the packet experiences the worst latency.) It has to wait the whole burst's service (or synonymously transmission) time. At the 2nd node, the slot is long enough to just accommodate the maximum burst, and is perfectly synchronized with the 1st node. But the arrival of the worst packet is just behind all the burst from the other input port. At the 3rd node's upper output port, the slot length can be as short as the 1st node. The worst packet's service time is at the last of the slot. At the 4th node, the slot can be as short as the twenty C flows' burst transmission time. The worst packet is served last within the burst.

As we have seen the worst case scenario in Figure 8, even with the perfect slot assignment, the latency bound is 1.712ms.

Similar arguments can be applied to the worst case analysis for the audio or video flows. Their latency upper bounds are identical at 2.032ms.

Now, let us consider the case where the C&C flows and their queue have a higher priority than the A or V traffic. Assume that the strict priority scheduler can preempt the lower priority packets. Then the C&C flows are treated as if there is no other traffic. The slot length can be as small as the twenty C flows burst at 1st and 3rd, and 4th node, and the forty C flows burst at 2nd node. The latency bound in this case is 0.24ms.

B. LATENCY BOUND WITH THE DIFFSERV

The DiffServ framework uses the class-based queuing and strict priority scheduling. It also uses resource reservation and admission control for high priority traffic, such that the link capacity at every output port does not exceed the total input arrival rate of that port. In the studied case, the total arrival rate is well below the link capacity. It is 5.4% of the link capacity. The TSN, and the original 802.1Q itself is based on the DiffServ architecture implemented in Layer 2 network.

We assume for simplicity, that the three types of traffic share a single queue. If they are placed in separate queues, and if C&C flows have higher priority than A/V flows, the perfor-

mance of the flow under observation would be much better. We further assume, as in the TSN synchronous case, the lower priority traffic is preempted, again for analysis simplicity. By these assumptions it is equivalent to the case that the scheduler has only a FIFO queue. If the queues are separated according to the traffic types, the C&C flows would be served with a higher priority, thus have a lower latency.

The worst case in strict priority scheduler, or any other work-conserving packet scheduler, happens when the maximum burst of packets from all the flows arrive almost simultaneously, and the packet under observation arrives last among them. This is the case analyzed in Figure 9.

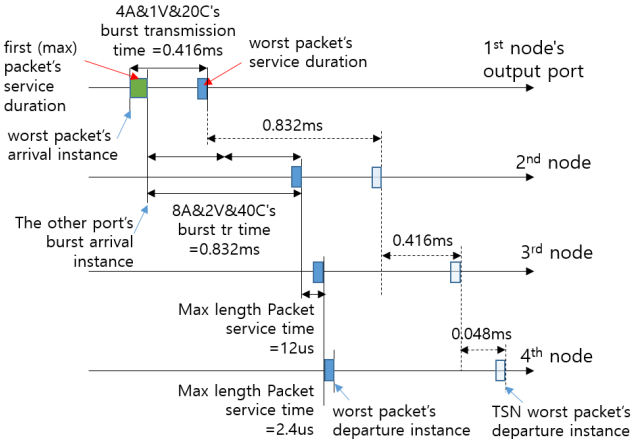


FIGURE 9. The worst case scenario of C&C flows with strict priority scheduling. The latency is 0.8584ms in this case. The dotted lines and light blue boxes denote the worst case of TSN synch. approach in Figure 8.

At the 1st node, the maximum burst from all the flows arrives at once. The worst packet's arrival is just a little behind all the other bursts. It has to wait the whole burst's service (or transmission) time. At the 2nd node, the service starts immediately after the first packet's service completion at the first node. This is natural with a work-conserving scheduler. Therefore, we should make the 2nd node's service start instance to be as late as possible to build the worst scenario. This happens when the first packet is the maximum length packet. The arrival of the maximum burst from the other node happens at the same time. Therefore, the worst packet is served last among the packets from the two input links. Considering the 1st and 2nd nodes together, the combined worst latency through two nodes is (the max length packet + total burst) service time.

At the 3rd node's upper output port, there is no more crossing traffic with the flow under observation. We argue that any packet in this case, after arrival, is served within the maximum length packet's service interval, which is 1500byte video packet service interval. Consider a minimal sized packet that arrives at the 3rd node right after the maximum length packet, without input link idle. We can think the arrival instances of both packets are almost identical. The minimal sized packet's service starts at the instance the max length packet's service completion. The service completion of the minimal packet is also almost identical to the max packet's service completion. Therefore, the minimal packet has also been delayed as long as the max packet's service interval. A similar argument applies to a packet with any length. We omit the argument for the readability of the paper. At the 4th node, there are only C&C flows, which have the same fixed length packets. The packets depart exactly as they arrived, because the transmission times of a packet in the different links are the same.

As we have seen the worst case scenario in Figure 9, with the strict priority scheduling with preemption, the latency bound is 0.8584ms, which is less than the one with the TSN synchronous approach. Similar arguments can be applied to the worst case analysis for the audio or video flows. Their latency upper bounds are identical at 0.868ms.

Now consider the case where the C&C flows have a higher priority than the A and the V traffic. Assume that the strict priority scheduler can preempt the lower priority packets. Then the C&C flows are treated as if there is no other traffic. With a similar argument with the one in Figure 9, we obtain the latency bound 0.1032ms.

As it is now clear, that the inferiority of the TSN synchronous approach in the case study comes from the fact that the slot assignment is on the burst level, i.e. every node should wait until the slot, or max burst, service of the previous node is finished. In the strict priority scheduling, the next node may have to wait at most the maximum length packet's service time. Considering this fact, it may be possible to improve the synchronous approach with a finer (e.g. packet) granularity slots, if the network topology and traffic pattern are simple enough for the schedulability test to be feasible.

C. LATENCY BOUND WITH THE FAIR

The core of the FAIR framework is the aggregation of flows based on input/output port pair. It may have more aggregation criteria, such as the traffic specification and required performance, but in this study the aggregation based on ports pair also makes the same traffic types to be aggregated into a FA. Since the FAs do not need to be segregated, the minimal IRs are not necessary in this topology.

Note that the separation of FAs for different traffic types, thus separation of queues, does not imply the priority differentiation among the traffic types. The queues are served fairly and accordingly to its total arrival rates. In this study we employ the deficit round robin (DRR) scheduler at every node. DRR is relatively simpler than the ones with 'virtual service time' based schedulers such as the packetized generalized processor sharing (PGPS) scheduler, yet the latency performance is acceptable. It is also known to preserve the fair share of the competing queues [48].

The graphical analysis in the previous subsections does not work well with the DRR. Instead we adopt the notion of Latency-Rate (LR) server [49] and its known properties. The DRR is the representative round-robin LR scheduler.

The four nodes that the flow under observation passes through, can be considered as a single scheduler with the total *latency* that is the sum of the *latencies* of all the four

nodes, according to the LR server property. This property is interpreted as ''pay burst only once''. Table 6 lists the symbols used in this analysis.

TABLE 6. Mathematical symbols in the case study.

Symbol	Quantity
L_i	Maximum packet length of flow i
L_{max}	Maximum packet length of all the flows in a scheduler
r	Link capacity
ρ_i	Arrival rate of flow i
σ_i	Maximum burst size of flow i
φ_i	Quantum value assigned for flow i
Θ_i^S	Latency of flow i at scheduler S

Formally stating, if the flow i traverses only the LR schedulers S_j in its path (with total k LR schedulers), then the end-to-end latency experienced by the packets in the flow is bounded by the following inequality [49].

$$D_i \leq \frac{\sigma_i - L_i}{\rho_i} + \sum_{j=1}^{k} \Theta_j^{S_j}. \tag{1}$$

The *latency* (written in italic) defined in LR server concept is a different term used elsewhere in this work. Latency in non-italic is synonymous to delay. *Latency* of a scheduler in LR server can be interpreted to be a maximum interval a flow may have to wait, from the start of a busy period, to be served with its allocated service rate. The packetized generalized processor sharing (PGPS) is an ideal but complex LR scheduler. PGPS's *latency* is given as follows [49].

$$\Theta_i^{PGPS} = \frac{L_i}{\rho_i} + \frac{L_{max}}{r}. \tag{2}$$

The *latency* of a DRR scheduler, in case the quantum values may be smaller than the max packet length, is given as follows [50].

$$\Theta_i^{DRR} = \frac{1}{r}\left[(F - \varphi_i)\left(1 + \frac{L_i}{\varphi_i}\right) + \sum_{n=1}^{N} L_n\right], \tag{3}$$

where F is the sum of all quantum values (φ_i) of active flows in the scheduler, and N is the number of active flows. Quantum refers to the amount of data serviced at one time, which is determined in proportion to the service rate allocated to each flow [48]. Note that the *latency* of DRR is a function of the sum of all the flows' max packet lengths, while the *latency* of PGPS is only affected by the maximum packet length overall. Therefore, the *latency* of DRR is affected by the number of the other flows. It would be preferable to use the PGPS in cases where the complexity of PGPS is acceptable.

In this case study we assume that the flows with the same traffic type and the same input/output port are aggregated into s single FA. There are six such FAs in the network for the case study. An FA is allocated to a separated queue. The queues in a node are served by DRR with the input rates proportional to the quantum values; 128bit for the four flows' Audio FA (total 6.4Mbps), 220bit for the single flow's Video FA (11Mbps), and 192bit for the twenty flows' C&C FA (total 9.6Mbps). The sum of the quantum values, F in (3), is 540 for example in the 1st node. Solving (3) for each node, we get $\Theta_i^{1st\ node} = 21.1$us, $\Theta_i^{2nd\ node} = 44.8$us, $\Theta_i^{3rd\ node} = 21.1$us, and $\Theta_i^{3rd\ node} = 2.4$us. From (1), we get the latency bound to be 0.2164ms. Note that for the C&C FA the max burst size is equal to the twenty times of the max packet length.

Similar arguments can be applied to the worst case analysis for the audio or video flows. Their latency upper bounds are 0.1391ms and 1.024ms, respectively. Note that the bound of audio flows is lower than that of C&C flows, since they have less number of flows and less amount of maximum burst within a flow aggregate. The bound of the video flows is larger, again because of its larger maximum burst. It is noteworthy that the FAIR framework isolates the flows from the other types of flows and their bursts, with separated queues per FA. Therefore, the FAIR is able to differentiate the latency bounds for each flow type. Observe that (1) or (3) does not include the max burst term of the other types of flows.

The three frameworks' latency bounds for the C&C flow is summarized in Table 7. Among the solutions with the same priority assignments to A, V, and C traffic, the FAIR framework's upper bound is the lowest. The results in Tables 7 and 8 are accurate since they are from the exact analysis. The latency bounds provided by all three solutions are within the required deadline of 5ms. Giving higher priority to C&C flows greatly helps reduce the latency bounds both for the TSN and the strict priority scheduling.

TABLE 7. Latency bounds for C&C flows with different frameworks.

Framework	Latency bound for C&C flow in the case study
TSN Synchronous approach	1.712ms
TSN Synchronous, with C&C flows having higher priority	0.24ms
Strict priority scheduling	0.8584ms
Strict priority scheduling, with C&C flows having higher priority	0.1032ms
FAIR	0.2164ms

The three frameworks' latency bounds for each type of flows are summarized in Table 8. The audio and video flows' bounds by all solutions are also within the required deadlines.

D. JITTER CONTROL WITH EACH SOLUTION

We consider the problem of guaranteeing a jitter bound with the network solutions in the case study. For the C&C flow, either the FAIR or the DiffServ solution can be the network of choice.

The FAIR can guarantee 0.2164ms latency upper bound. Therefore, the parameter U is equal to 0.2164ms. The sum of

TABLE 8. Latency bounds for Each flows with different frameworks.

Framework	C&C flow	Audio flow	Video flow
TSN Synchronous	1.712ms	2.032ms	2.032ms
Strict priority	0.8584ms	0.868ms	0.868ms
FAIR	0.2164ms	0.1391ms	1.024ms

the packet transmission time in four nodes is 4 times 2.4μs, which is the latency lower bound, the parameter W is equal to 9.6μs. If we let the parameter m in Section III to be equal to U, 0.2164ms, then from Theorem 1, the E2E buffered latency upper bound is (m+U-W), which is equal to 0.4304ms and the jitter bound is U-m, is equal to 0. We can achieve a perfect synchronization without time-synchronization functions specified in the TSN standard, while meeting the latency bound requirements (10ms in this case) by a wide margin.

Note that CPRI is expected to maintain the maximum end-to-end latencies less than 100-250μs [5], [6], jitter within 65-100ns [6], [7], in 10G Ethernet network. Scaling down to a 1G network, we infer about 1$\sim$2.5ms latency bound and 0.65$\sim$1μs jitter bound. If we let the jitter bound to be 1μs, then again from Theorems 1 and 3, with the FAIR framework, m = U-1us = 0.2154ms, m+U-W = 0.4294, U-m = 1us. As such the latency bound 0.4294ms meets even the stringent CPRI requirement of 1ms.

The DiffServ framework, with simple strict priority schedulers for high priority traffic with preemption, can guarantee 0.8584ms. We can also achieve zero jitter with the DiffServ framework, with a buffer, at the cost of lengthening the E2E buffered latency upper bound roughly twice, to 1.7144ms. This is the similar bound to that of the TSN synchronous approach, which is 1.712ms. With 1us jitter bound, the buffered latency bound is 1.7134ms. See Table 9 for comparison.

In Table 9, the latency bound of TSN is E2E latency bound. All the other bounds for strict priority and the FAIR frameworks are the E2E buffered latency bounds.

E. LATENCY BOUND OF THE FAIR WITH VARYING TOTAL INPUT RATE

The FAIR framework can provide latency bounds also in highly utilized situations. If we let the combination of 8 audio flows, 2 video flows, and 40 C&C flows be a unit of input traffic, and increase the number of the units, we can examine the relationship of the traffic load and the latency bound. From (1) and (3), we find the E2E latency is equal to (p$*$154.4K + 61.3K)/1G second, where p is the number of the traffic unit. With p = 1, which is the case studied in previous subsections, the link utilization is 5.4%. The E2E latency of the FAIR framework versus the link utilization is shown in the Figure 10. Roughly twice the E2E latency value in this graph is the E2E buffered latency with zero jitter. We can see with the p value is equal to 15, or the utilization around 81%, it is still possible to guarantee 5ms E2E buffered latency with zero jitter.

TABLE 9. Latency and Jitter bounds for C&C flows in the case study with the proposed framework.

Framework	Latency bound	Jitter bound
Strict priority with buffer set to zero jitter	1.7144ms	0
Strict priority with buffer set to 1us jitter	1.7134ms	1us
FAIR with buffer set to zero jitter	0.4304ms	0
FAIR with buffer set to 1us jitter	0.4294ms	1us
TSN Synch. approach (without buffer)	1.712ms	-

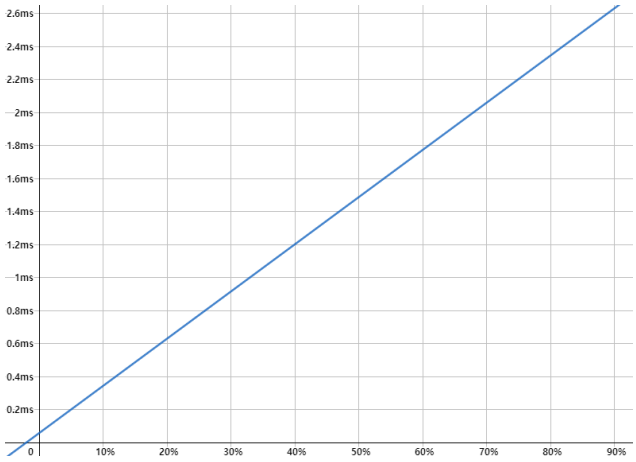


FIGURE 10. E2E latency bound of C&C flows with the FAIR framework in the case study network, versus link utilization. Up to 80% load, the E2E buffered latency is less than 5ms, the C&C flows' latency requirement.

VI. SIMULATIONS

We simulate the network of Figure 6 that is also used for the case study in Section V. The input traffic and the routes also follow the description in Figure 6. The flows' characteristics follow Table 5. The simulation environment is constructed using OMNeT++. It is a C++ based network simulation framework, which is known to be good for constructing discrete event network simulations [51]. The detailed simulation parameters for input data generation are given in Table 10. Note that the packet length, burst size, and the period are all fixed values.

TABLE 10. Input data parameters used in the simulation.

Traffic type	{Number of flows, Packet length, Burst size per flow, Burst generation period, Number of total packets generated}
Audio	{8, 2Kbit, 2Kbit, 1.25ms, 105.6K}
Video	{2, 12Kbit, 360Kbit, 33ms, 30K}
Command & Control	{40, 2.4Kbit, 2.4Kbit, 5ms, 132K}

Also given in Table 11 is the DRR scheduler's parameters used for the FAIR solution. The capacities for all the links are set to 1Gbps.

TABLE 11. DRR scheduler's parameters used for the FAIR solution in the simulation.

Traffic type	Number of flows in an FA	Number of FAs (queues)	Quantum size for queue
Audio	4	2	128
Video	1	2	220
C&C	20	2	192

In the simulation, all the flows have the same phase, i.e. emit the maximum burst at the start of the simulation. It is so set in order to observe the worst case behavior at the earlier stage of the simulation. A single run of the simulation lasts for 16.5 seconds. The C&C flows, the audio flows, and the video flows produce 3,300, 13,200, and 15,000 packets respectively, for a single run. We perform 100 runs independently. All C&C flows with the same input port and output port were observed.

Figures 11 to 13 depict the E2E latency bounds obtained in the previous section, and the latency distributions obtained from the simulations of the C&C, audio, and video flows with three different solutions. With all the solutions, the theoretical latency bounds match to the maximum observed latencies, except the video flows' bound in the FAIR solution. This is because the latency upper bound in (1) considers an even worse scenario than the case study. In the case study with the FAIR, some of the video packets within a burst remain in the queue even after all the other types of packets left the node. The bound in (1) assumes the continuous backlog of all the queues. In our case study, the later packets in a video burst do not see the backlog of other types of flows in the scheduler. As such the worst video packet experiences less latency than the theoretical bound suggests. Other than this case all the theoretical bounds match to the simulation results.

In Figure 11, in the box plot of TAS, the largest latencies around 1.4∼1.75ms are observed when the three types of flows' slots overlap. Also in the box plot of TAS, the dots in the middle, around 0.6∼0.7ms, represent the latencies of the packets in the slots that overlap partly with video flow's slot. The TAS would show a smaller jitter compared to the FAIR or the DIffServ, if these outliers were not present. This is because for C&C flows, the period 5ms is the multiple of the period of audio flows 1.25ms. Therefore, the C&C flows' bursts always overlap with that of audio. This regularity contributes to the smaller jitter of the TAS. Note that the average latency of the TAS is also much larger than the others.

In Figure 12, the FAIR shows the smallest bound. The FAIR, or the DRR scheduler, protects the audio flows from the burst of other flows. The FIFO scheduler in the DiffServ also works well, except when the bursts of the audio and the other type of flows overlap. The TAS shows the larger jitter and the larger latency bound. This is because the audio flows,

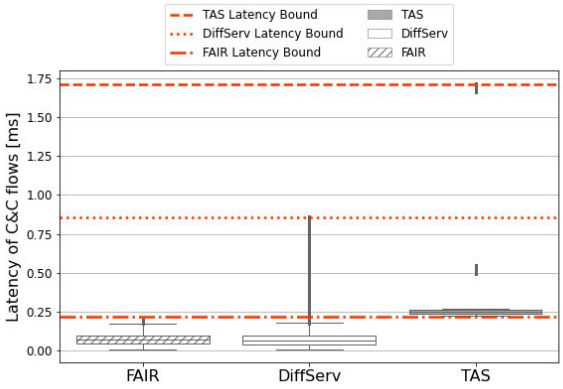


FIGURE 11. Theoretical E2E latency bounds and the latency distribution from simulation in box-and-whisker plots, of C&C flows with three different solutions in the case study network.

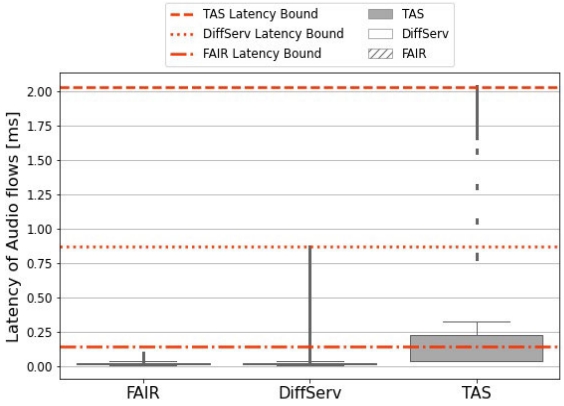


FIGURE 12. Theoretical E2E latency bounds and the latency distribution from simulation of audio flows.

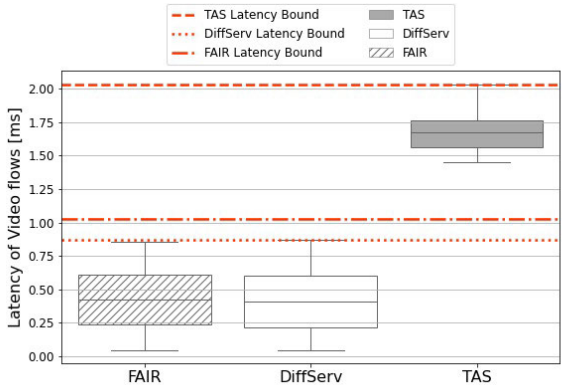


FIGURE 13. Theoretical E2E latency bounds and the latency distribution from simulation of video flows.

with the shortest period of 1.25ms, overlap with the other types of flows with roughly 1/4 probability.

In Figure 13, we can see that the jitters of video flows in all three solutions are higher than those of the other flows. Also the latency distributions of the FAIR and the DiffServ are almost identical. This is because a video flow has a much larger burst of 360Kbit. The other flows' bursts do not significantly affect the latency of video flows.

Figure 14 depicts E2E buffered latencies of all the packets from two random sample C&C flows, that are set to have zero jitter, observed during the simulations. According to the actual latency experienced by the first packet of a flow, the latencies of the subsequent packets are decided. Note that the fixed latencies of the sample flows are placed around the E2E latency bound, 0.2164ms. It is shown that the flows achieve the zero latency with the proposed framework.

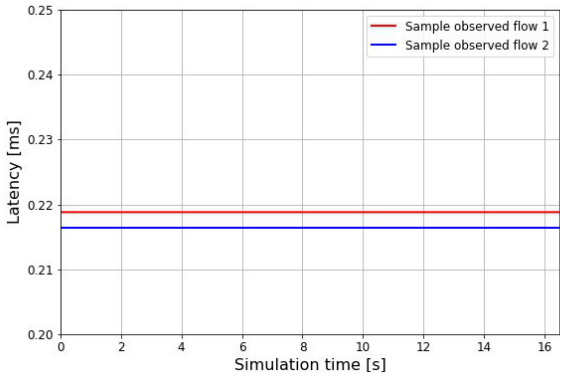


FIGURE 14. E2E buffered latencies of all the packets of two random sample C&C flows observed during the simulation.

VII. CONCLUSION

We have proposed a novel framework that guarantees any desired amount of jitter bound, without time-synchronization. The network in the framework can be of any size, any topology, or any input pattern, as long as it guarantees latency bounds. We have shown that, with a proper selection of the latency guarantee network, such as the FAIR, the proposed framework can achieve a better latency bound, even with zero jitter, than that of the TSN Synchronous approach. This work seriously challenges the necessity of the tremendous effort for time-synchronization, slot assignment, and coordination among the nodes. The proposed framework, however, requires the additional buffer at the egress of a network, per flow buffering operation, and time-stamping every packet. These functions impose additional complexity. We argue that such a cost for the additional functions is negligible compared to the network-wide time-synchronization and slot scheduling functions that are required in TSN.

The insight we have gained through the analysis is that the isolation of critical packets or flows from the bursts of other traffic is essential for latency and jitter minimization. In a slotted operation, this means a smaller slot size, around a packet transmission time, is preferable. In traditional statistical multiplexing environments, this means a flow or FA level queuing and scheduling are preferable.

The basic philosophy behind the proposed framework is to minimize the latency first by taking advantage of the work-conserving schedulers, and then adjust the inter-departure times to reproduce the inter-arrival times, at the boundary of a network. We argue that this is simpler than trying to minimize the latency and the jitter at the same time. The direct benefit of such simplicity is its scalability. The time-synchronization would be useful for wireless networks for energy efficiency and multiple access, where synchronized channel access reduces the collision rate. In wired full-duplex links, for the synchronized approach to be beneficial, a more careful slot planning would be necessary.

VIII. FUTURE WORKS

It is for further study how we can execute the buffer operation precisely and efficiently in the proposed framework. Note that, in order to have a packet depart the buffer at a certain instance, the transmission of the packet must have started earlier as much as its transmission time. Moreover, the buffer-out instance c_n can be the same with the buffer-in instance b_n, according to Algorithm 1. Therefore, we need a 'look-ahead' capability to decide the incoming packet should be cut-through, which is similarly required to the IRs in the TSN ATS and the FAIR. Note that the look-ahead capability is also required to the TAS in order for a transmitted packet not to exceed the slot boundary. While the implementation is feasible, how efficient could it be should be investigated in future work. A tight cooperation with the network and the buffer would be necessary in this regard. A buffer embedded in the network boundary node can solve it easier. On the other hand, in cases that look-ahead and cut-thorough are too complex to implement, simply letting additional buffer latency could solve the problem as well.

The increased E2E buffered latency bound by the proposed framework, from U to almost 2U, can be mitigated by one of some added functionalities. 1) First, one can measure the latency of the flows' first packet in the network exactly, and buffer it to make its E2E buffered latency to be U. Then every subsequent packet will experience the same E2E buffered latency that is U, with zero jitter. How one can measure the latency is for further investigation. For example, one can synchronize only the boundary nodes of the network. 2) Second, one can expedite the first packet's service with a special treatment, to make its latency lower than the other packets of the flow. If we can make the first packet's latency to be a small value Δ, then every packet will experience the same buffered latency $\Delta+U$, with zero jitter. Considering that the E2E latency bound is obtained from the case in which rare events occur simultaneously, however, the first packet's latency is likely to be far less than what the bound suggests. Therefore, the special treatment to the first packet may be ineffective in real implementations.

The proposed framework can also be used for jitter control among multiple sources' flows having a single destination. When a session is composed of more than one sources, physically or virtually separated, the buffer at the boundary can mitigate the latency variations of packets from different sources due to different routes or network treatments. Such a scenario may arise in cases such as 1) that a central unit controls multiple devices for a coordinated execution in smart factories, or 2) multi-user conferencing applications, in which multiple devices/users physically separated can have a difficulty in real-time interactions [10]. The sources, or the ingress

boundary nodes of the network, need to be synchronized with each other in order for the time-stamps from separated sources to be able to identify the absolute arrival instances, however. The detailed operations of such multi source scenarios are to be investigated.

Finally, as we have learned that a smaller slot would certainly improve the TSN synchronous approach, it is to investigate the possibility of a centralized coordinated operation of a network with very small fixed-length slots. If we fragment packets into a fixed-length 'cell', e.g. 64byte, and make all the slots in the network to be of length exactly a cell transmission duration, and reassemble the cells only at the boundary of a network, then the whole network could be seen as a single multi-stage cell switch. Such a switch design problem has been extensively studied for asynchronous transfer mode (ATM) switches in 90's [52], [53] and for data center networks (DCNs) recently, where high throughput and scalability are the primary design objectives [54], [55]. The topology of a deterministic network would become one of the design parameters. With the help from the centralized SDN and the reinforcement learning paradigm in the networking area, an intelligently synchronized slot approach based on such cell switch architectures would also be promising.

REFERENCES

[1] L. Lachello, P. Wratil, and A. Meindl, "Industrial Ethernet facts," Ethernet Powerlink Standardization Group, Fredersdorf, Germany, Tech. Rep., 2017.

[2] D. Cavalcanti, J. Perez-Ramirez, M. M. Rashid, J. Fang, M. Galeev, and K. B. Stanton, "Extending accurate time distribution and timeliness capabilities over the air to enable future wireless industrial automation systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1132–1152, Jun. 2019, doi: 10.1109/JPROC.2019.2903414.

[3] A. M. Romanov, F. Gringoli, and A. Sikora, "A precise synchronization method for future wireless TSN networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3682–3692, May 2021, doi: 10.1109/TII.2020.3017016.

[4] P. J. Chaine, M. Boyer, C. Pagetti, and F. Wartel, "TSN support for quality of service in space," presented at the 10th Eur. Congr. Embedded Real Time Softw. Syst. (ERTS), Toulouse, France, Jan. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02441327/document

[5] *Study on New Radio Access Technology: Radio Access Architecture and Interfaces*, document TR 38.801, 3GPP, Version 14.0.0, 2017. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3056

[6] T. Wan and P. Ashwood-Smith, "A performance study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv enhancements," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, San Diego, CA, USA, Dec. 2015, pp. 1–6, doi: 10.1109/GLOCOM.2015.7417599.

[7] D. Chitimalla, K. Kondepu, L. Valcarenghi, M. Tornatore, and B. Mukherjee, "5G fronthaul-latency and jitter studies of CPRI over Ethernet," *J. Opt. Commun. Netw.*, vol. 9, no. 2, pp. 172–182, Feb. 2017, doi: 10.1364/JOCN.9.000172.

[8] *Common Public Radio Interface (CPRI); Interface Specification*, CPRI Specification, Version 7.0, 2015. [Online]. Available: http://www.cpri.info/downloads/CPRI_v_7_0_2015-10-09.pdf

[9] M. Waqar, A. Kim, and P. K. Cho, "A transport scheme for reducing delays and jitter in Ethernet-based 5G fronthaul networks," *IEEE Access*, vol. 6, pp. 46110–46121, 2018, doi: 10.1109/ACCESS.2018.2864248.

[10] *ITU-T Y.3000-Series—Network 2030 Services: Capabilities, Performance and Design of New Communication Services for the Network 2030 Applications*, document Rec. ITU-T Y.Sup66, International Telecommunication Union, Geneva, Switzerland, Jul. 2020.

[11] *System Architecture for the 5G System (5GS); Stage 2 (Release 16)*, document TS 23.501, 3GPP, 2020. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144

[12] *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)*, document IETF RFC 3393, Nov. 2002. [Online]. Available: https://www.rfc-editor.org/info/rfc3393

[13] *Time-Sensitive Networking Task Group*. Accessed: Mar. 07, 2021. [Online]. Available: https://www.ieee802.org/1/pages/tsn.html

[14] *Timing and Synchronization Aspects in Packet Networks*, document Rec. ITU-T G.8261/Y.1361, International Telecommunication Union, Geneva, Switzerland, Aug. 2019.

[15] S. C. Ergen and P. Varaiya, "TDMA scheduling algorithms for wireless sensor networks," *Wireless Netw.*, vol. 16, no. 4, pp. 985–997, May 2010, doi: 10.1007/s11276-009-0183-0.

[16] J. Mao, Z. Wu, and X. Wu, "A TDMA scheduling scheme for many-to-one communications in wireless sensor networks," *Comput. Commun.*, vol. 30, no. 4, pp. 863–872, Feb. 2007, doi: 10.1016/j.comcom.2006.10.006.

[17] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," in *Proc. 24th Int. Conf. Real-Time Netw. Syst.*, 2016, pp. 203–212.

[18] M. L. Raagaard, P. Pop, M. Gutiérrez, and W. Steiner, "Runtime reconfiguration of time-sensitive networking (TSN) schedules for fog computing," in *Proc. IEEE Fog World Congr. (FWC)*, Santa Clara, CA, USA, Oct. 2017, pp. 1–6, doi: 10.1109/FWC.2017.8368523.

[19] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "An SDN architecture for time sensitive industrial IoT," *Comput. Netw.*, vol. 186, Feb. 2021, Art. no. 107739, doi: 10.1016/j.comnet.2020.107739.

[20] *Network Time Protocol Version 4: Protocol and Algorithms Specification*, document IETF RFC 5905, Jun. 2010. [Online]. Available: https://www.rfc-editor.org/info/rfc5905

[21] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588, 2019.

[22] O. Al-Kofahi, "Evaluating time synchronization using application-layer time-stamping," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6, doi: 10.1109/wcnc.2016.7564909.

[23] T. Kovácsházy and B. Ferencz, "Performance evaluation of PTPd, a IEEE 1588 implementation, on the ×86 Linux platform for typical application scenarios," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf.*, Graz, Austria, May 2012, pp. 2548–2552, doi: 10.1109/I2MTC.2012.6229387.

[24] G. Cena, M. Cereia, I. C. Bertolotti, S. Scanzio, A. Valenzano, and C. Zunino, "A software implementation of IEEE 1588 on RTAI/RTnet platforms," in *Proc. IEEE 15th Conf. Emerg. Technol. Factory Automat. (ETFA)*, Bilbao, Spain, Sep. 2010, pp. 1–8, doi: 10.1109/ETFA.2010.5640955.

[25] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Synchronization quality of IEEE 802.1AS in large-scale industrial automation networks," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Pittsburgh, PA, USA, Apr. 2017, pp. 273–282, doi: 10.1109/RTAS.2017.10.

[26] T. H. Szymanski, "Bounds on end-to-end delay and jitter in input-buffered and internally-buffered IP networks," in *Proc. IEEE Sarnoff Symp.*, Princeton, NJ, USA, Mar. 2009, pp. 1–7, doi: 10.1109/SARNOF.2009.4850287.

[27] T. H. Szymanski, "A low-jitter guaranteed-rate scheduling algorithm for packet-switched IP routers," *IEEE Trans. Commun.*, vol. 57, no. 11, pp. 3446–3459, Nov. 2009, doi: 10.1109/TCOMM.2009.11.070666.

[28] T. H. Szymanski and D. Gilbert, "Internet multicasting of IPTV with essentially-zero delay jitter," *IEEE Trans. Broadcast.*, vol. 55, no. 1, pp. 20–30, Mar. 2009, doi: 10.1109/TBC.2008.2007455.

[29] *RTP: A Transport Protocol for Real-Time Applications*, document IETF RFC 3550, Jul. 2003. [Online]. Available: https://www.rfc-editor.org/info/rfc3550

[30] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: Architecture and mechanism," in *Proc. Conf. Commun. Archit. Protocols (SIGCOMM)*, Baltimore, MD, USA, Aug. 1992, pp. 14–26, doi: 10.1145/144179.144199.

[31] *Integrated Services in the Internet Architecture: An Overview*, document IETF RFC 1633, Jun. 1994. [Online]. Available: https://www.rfc-editor.org/info/rfc1633

[32] *An Architecture for Differentiated Services*, document IETF RFC 2475, Dec. 1998. [Online]. Available: https://www.rfc-editor.org/info/rfc2475

[33] S. Abbasloo and H. J. Chao, "SharpEdge: An asynchronous and core-agnostic solution to guarantee bounded-delays," *CCF Trans. Netw.*, vol. 3, no. 1, pp. 35–50, Aug. 2020.

[34] H. Zhang and D. Ferrari, "Rate-controlled service disciplines," *J. High Speed Netw.*, vol. 3, no. 4, pp. 389–412, 1994.

[35] J. Joung, "Regulating scheduler (RSC): A novel solution for IEEE 802.1 time sensitive network (TSN)," *Electronics*, vol. 8, no. 2, p. 189, Feb. 2019.

[36] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched Ethernet networks," in *Proc. 28th Euromicro Conf. Real-Time Syst. (ECRTS)*, Toulouse, France, Jul. 2016, pp. 75–85, doi: 10.1109/ECRTS.2016.27.
[37] J.-Y. L. 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, doi: 10.1109/TNET.2018.2875191.
[38] J. Joung, "Framework for delay guarantee in multi-domain networks based on interleaved regulators," *Electronics*, vol. 9, no. 3, p. 436, Mar. 2020, doi: 10.3390/electronics9030436.
[39] *Framework for Latency Guarantee in Large Scale Networks Including IMT-2020 Network*, document Rec. ITU-T Y.3113, International Telecommunication Union, Geneva, Switzerland, Feb. 2021.
[40] Q. Li, D. Li, X. Jin, Q. Wang, and P. Zeng, "A simple and efficient time-sensitive networking traffic scheduling method for industrial scenarios," *Electronics*, vol. 9, no. 12, p. 2131, Dec. 2020, doi: 10.3390/electronics9122131.
[41] W. Quan, W. Fu, J. Yan, and Z. Sun, "OpenTSN: An open-source project for time-sensitive networking system development," *CCF Trans. Netw.*, vol. 3, no. 1, pp. 51–65, Sep. 2020, doi: 10.1007/s42045-020-00029-8.
[42] N. Navet and J. Migge, "Insights into the performance and configuration of TCP in automotive Ethernet networks," in *Proc. IEEE Standards Assoc. (IEEE-SA), Ethernet IP Automot. Technol. Day*, London, U.K., Oct. 2018, pp. 1–24.
[43] T. L. Mai, N. Navet, and J. Migge, "A hybrid machine learning and schedulability analysis method for the verification of TSN networks," in *Proc. 15th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Sundsvall, Sweden, May 2019, pp. 1–8, doi: 10.1109/WFCS.2019.8757948.
[44] N. Navet, J. Seyler, and J. Migge, "Timing verification of real-time automotive Ethernet networks: What can we expect from simulation?" in *Proc. 8th Eur. Congr. Embedded Real Time Softw. Syst. (ERTS)*, Toulouse, France, Jan. 2016, pp. 1–13. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01292312
[45] N. Navet, T. L. Mai, and J. Migge, "Using machine learning to speed up the design space exploration of Ethernet TSN networks," Univ. Luxembourg, Luxembourg City, Luxembourg, Tech. Rep., 2019. [Online]. Available: http://hdl.handle.net/10993/38604
[46] J. Migge, J. Villanueva, N. Navet, and M. Boyer, "Insights on the performance and configuration of AVB and TSN in automotive Ethernet networks," in *Proc. Embedded Real-Time Softw. Syst. (ERTS)*, Toulouse, France, Jan. 2018, pp. 1–10. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01746132
[47] N. Navet, J. Villanueva, J. Migge, and M. Boyer, "Experimental assessment of QoS protocols for in-car Ethernet networks," in *Proc. IEEE Standards Assoc. (IEEE-SA) Ethernet IP Automot. Technol. Day*, San Jose, CA, USA, Oct. 2017, pp. 1–33.
[48] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375–385, Jun. 1996, doi: 10.1109/90.502236.
[49] D. Stiliadis and A. Varma, "Latency-rate servers: A general model for analysis of traffic scheduling algorithms," *IEEE/ACM Trans. Netw.*, vol. 6, no. 5, pp. 611–624, Oct. 1998, doi: 10.1109/90.731196.
[50] L. Lenzini, E. Mingozzi, and G. Stea, "Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers," *IEEE/ACM Trans. Netw.*, vol. 12, no. 4, pp. 681–693, Aug. 2004, doi: 10.1109/TNET.2004.833131.
[51] *OMNeT++*. (2020). [Online]. Available: https://omnetpp.org/download/
[52] R. Y. Awdeh and H. T. Mouftah, "Survey of ATM switch architectures," *Comput. Netw. ISDN Syst.*, vol. 27, no. 12, pp. 1567–1613, 1995, doi: 10.1016/0169-7552(94)00081-4.
[53] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188–201, Apr. 1999, doi: 10.1109/90.769767.
[54] F. Hassen and L. Mhamdi, "A scalable multi-stage packet-switch for data center networks," *J. Commun. Netw.*, vol. 19, no. 1, pp. 65–79, Feb. 2017, doi: 10.1109/JCN.2017.000009.
[55] F. Hassen and L. Mhamdi, "A new paradigm to build scalable packet-switches for data center networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–7, doi: 10.1109/GLOCOM.2018.8647295.

JINOO JOUNG received the B.S. degree in electronics engineering from the Korea Advanced Institute for Science and Technology, Daejeon, South Korea, in 1992, and the M.S. and Ph.D. degrees in electrical and electronics engineering from New York University, New York City, NY, USA, in 1994 and 1997, respectively. He worked with Samsung Electronics and the Samsung Advanced Institute for Technology, from 1997 to 2005. His work at Samsung, includes various network SOC research and developments, especially for general packet radio service for 3G mobile systems, network processors for high-speed IP routers, and mobile application processors for smart hand-held devices. In 2005, he joined Sangmyung University, Seoul, South Korea. His research interests include network SOCs, high-speed network processing, switch architecture, network calculus, network QoS control, and autonomous wireless network scheduling/routing. He is active in international standardization activities. He is the main editor of various ITU-T Recommendations. He currently holds the TTA ICT International Standard Expert title.

JUHYEOK KWON received the B.S. degree in human-centered artificial intelligence from Sangmyung University, Seoul, South Korea, in 2020, where he is currently pursuing the M.S. degree in intelligence information engineering. His research interests include deterministic network services, wireless autonomous networks, and network SOC design.