IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 February 2023, accepted 3 March 2023, date of publication 10 March 2023, date of current version 16 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3255416

RESEARCH ARTICLE

Incremental Routing and Scheduling Using Multipath and Nonzero Jitter Bound for IEEE 802.1 Qbv Time Aware Shaper

DONG-JUN LEE, (Member, IEEE)
School of Electronics and Information Engineering, Korea Aerospace University, Goyang, Gyeongi 10540, South Korea
e-mail: ldj@kau.ac.kr

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) Grant funded by the Ministry of Science and ICT (MSIT) of the Korea Government under Grant 2018-0-00846, and in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF2018R1D1A1B07048319.

ABSTRACT IEEE 802.1 Qbv time-aware shaper (TAS) has been developed for delivery of time-critical periodic frames of time-triggered (TT) flows over Ethernet under strict end-to-end latency and jitter bounds. Efficient routing and flow scheduling are critical for successful deployments of TAS systems. Although many routing and flow scheduling schemes for TAS have been proposed so far, most of them are restricted to single path routing and zero jitter delivery. Multipath routing and nonzero jitter bound could provide more flexibility in scheduling thus increase scheduling success. However they have been rarely studied in TAS scheduling. In this work, online incremental routing and scheduling schemes utilizing multipath routing and nonzero jitter bound are proposed based on no-wait packet switching mechanism for TAS. Two routing and scheduling schemes with nonzero jitter bound are proposed, i.e., incremental single path routing and scheduling with nonzero jitter bound (ISPRS-NZJ) and incremental multipath routing and scheduling with nonzero jitter bound (IMPRS-NZJ). Both schemes are based on the proposed resource-centric scheduling approach in which information on feasible free time-slices in the paths for a TT flow is updated according to flow scheduling result and used for scheduling of next flow. Paths for a flow are searched by depth-first way and cumulative free time-slices (CFTSs) in each link of a path are found and used in ISPRS-NZJ. In IMPRS-NZJ, combined cumulative free time-slices (CCFTSs) are obtained from CFTSs and used for scheduling. Numerical results show that scheduling with nonzero jitter bound or multipath routing could reduce schedule failure rates compared with typical single path scheduling with zero jitter bound. Also it is shown that scheduling with nonzero jitter bound is more effective in reducing schedule failure rates than scheduling with multipath routing.

INDEX TERMS Jitter, multipath routing, QoS, real-time Ethernet, scheduling, time-aware shaper, time-sensitive network. 802.1Qbv.

I. INTRODUCTION

IEEE 802.1 Time-Sensitive Networking (TSN) defines IEEE 802.1 Qbv time-aware shaper (TAS) for delivery of time-triggered (TT) traffic over Ethernet. It is designed to satisfy strict end-to-end latency and jitter requirements for transmissions of periodic frames of time-critical flows [1], [2]. Required latency of a TT flow should be guaranteed by careful flow scheduling by which time slots are allocated to the flow in the links of a path [3]. Flow scheduling results are used to generate gate control list (GCL) which designates timings of periodic gate on-off operations of an egress port of a switch.

TSN application areas range from static small scale networks such as automotive networks [4], [5] to more dynamic large scale networks such as Industry 4.0 and crosshaul network for 5G ultra reliable low latency communications (URLLC) [6], [7], [8]. In dynamic TSN application

The associate editor coordinating the review of this manuscript and approving it for publication was Abdel-Hamid Soliman.

environments, a large number of TT flows are expected to be served and frequent rescheduling of flows could arise. Thus online scheduling is necessary in this situation.

Computation load of static TAS flow scheduling is known to be generally very high and thus scalability is one of critical problems which should be solved for successful employment of TAS networks. Due to high complexity, static TAS flow scheduling schemes known so far are rather restricted. One of well-known scheduling approaches for static scheduling cases is scheduling with zero jitter and nonzero queuing delay [9], [10]. In this approach, availability of time slots in links and first-in-first-out (FIFO) queue operation at each switch are considered in scheduling. Another important approach is flow scheduling with no-wait packet switching in which zero jitter and zero queuing delay are experienced by frames during delivery in network [11]. No-wait scheduling does not consider queue management at switches and thus results in fixed latency. These approaches can be modeled as Satisfiability Modulo Theory (SMT) or integer linear programming (ILP). Also, incremental scheduling schemes have been proposed to overcome high computation load of static scheduling problems. They could be also useful for online scheduling in which flows arrive and depart dynamically [12], [13].

TT flows are generally associated with end-to-end latency and nonzero jitter bound. Complexity of TAS scheduling could increase much when nonzero jitter bound is considered in scheduling. Thus most TAS scheduling schemes adopt zero jitter constraint and make no explicit consideration of nonzero jitter bound of a flow. With zero jitter transmission, most TAS scheduling schemes use a flow-centric approach in which the initial transmission time offset in a hyperperiod for periodic frames of a scheduled flow is registered and scheduled flows are examined one by one for scheduling of a new flow. In this approach, scheduling with nonzero jitter is difficult to apply since it could increase the computation load much more.

In Ethernet, multipath routing method such as shortest path bridging (SPB) has been used in forwarding of best-effort traffic for load balancing [14], [15], [16]. However, in routing and scheduling for TT traffic, multipath routing has been rarely considered and most of the research works on TAS routing and scheduling have been restricted to single path routing. Thus current research works on TAS routing and scheduling are lack of consideration of nonzero jitter bound and scheduling for multipath routing.

TAS flow scheduling exploiting nonzero jitter bound and multipath routing could provide more flexibility in scheduling and improve scheduling performance. Thus in this work, we propose new online incremental routing and scheduling schemes which exploit multipath routing and nonzero jitter bound. We propose a resource-centric scheduling approach in which free time-slices of a path are registered dynamically according to scheduling results and feasible free time-slices common in the links of a path for a flow are identified for scheduling under no-wait switching constraint. Feasible free time-slices common in a path for a flow are called cumulative free time-slices (CFTS) and time-slices in CFTS set of a path are allocated to the flow.

Based on the resource-centric approach, we propose two incremental routing and scheduling schemes for TT flows which could exploit nonzero jitter bound. One is incremental single path routing and scheduling with nonzero jitter bound (ISPRS-NZJ) scheme in which flows are incrementally scheduled using a shortest path of each flow. The other is incremental multipath routing and scheduling with nonzero jitter bound (IMPRS-NZJ) scheme which could use multiple shortest paths simultaneously for delivery of frames of a flow. In ISPRS-NZJ, CFTS set in the last hop link is used for flow scheduling. In IMPRS-NZJ, CFTSs of each incoming link of a node are combined resulting in combined cumulative free time-slice (CCFTS) with identification of the incoming link and flow scheduling is performed based on the CCFTS set of the destination node.

Numerical results show that scheduling with nonzero jitter bound results in reduced schedule failure rates both in ISPRS-NZJ and IMPRS-NZJ schemes. IMPRS-NZJ scheme shows schedule failure rates similar to ISPRS-NZJ scheme and shows better performance in rather limited cases. Running times of both schemes do not show much difference. The proposed schemes have $O(N^2)$ time complexity with N flows.

The remainder this paper is organized as follows. Section II explains the related work. Section III provides system model. Section IV elaborates the proposed ISPRS-NZJ scheme and section V the proposed IMPRS-NZJ scheme. Section VI provides numerical results. Finally, Section VII gives conclusion.

II. RELATED WORK

Many research works on TAS scheduling have assumed static scheduling situation in which a number of flows are considered simultaneously for time-slot allocation. Static flow scheduling has been studied with SMT which is NP-complete [9], [10], [17], [18], [19]. Most research works on scheduling assume zero jitter transmission. For deterministic frame transmission in FIFO queue of each switch, frame isolation constraint or queue order constraint are considered [9], [20].

High time complexity of static TAS scheduling could restrict schedulability of flows. Thus there exist research works using heuristic scheduling approaches in static TAS scheduling for reduced complexity [21], [22], [23], [24]. They use greedy approach in finding available transmission time offsets for flows. Multiple queues for time critical flows could be permitted for avoidance of frame isolation constraint [21]. Although heuristic methods improve scheduling running time compared with SMT-based methods, their time complexity analyses are generally not given explicitly and performance evaluations are rather restricted. Most TAS scheduling schemes have assumed zero-jitter delivery. Post processing on zero jitter scheduling result could be applied for latency reduction and thus nonzero jitter could result in [22]. But nonzero jitter bound has not been used for improvement of schedulability and the effect has not been studied explicitly. In [24], incremental scheduling scheme is proposed in which

transmission time of a flow is increased by basic time unit when collision occurs with previously scheduled flows.

Some research works model TAS scheduling problem as no-wait packet scheduling problem which could be formulated as NP-hard ILP problem [11], [25], [26], [27]. In no-wait scheduling, queuing delay is disregarded at a switch. Related works still adopt zero jitter transmission assumption.

Incremental scheduling has been applied for no-wait scheduling [11]. Scheduling procedure could be divided into sequencing of flows and incremental scheduling for each sequence of flows. For incremental no-wait scheduling, time-tabling method is proposed which has $O(N^3)$ time complexity for N flows [11]. As in most research works on TAS scheduling, a flow-centric method is used in this work. In [28], flow scheduling is coordinated with tasks generating data to transmit in end devices. Thus end-to-end latency bound is applied between tasks in end devices not just to delivery in network, while incremental scheduling with no-wait switching and zero jitter is still assumed for frame delivery in a network.

Incremental scheduling could be used in dynamic online scheduling situations, in which flow sequencing is unnecessary. In [13], online no-wait scheduling scheme is applied and ILP formulation is used for finding the earliest possible transmission start time offset of a flow in a hyperperiod. This approach is very similar to the result of time-tabling method in [11] with equal time complexity of $O(N^3)$.

To reduce computational complexity, stream partitioning approach has been applied to scheduling [27], [29]. In this approach, streams are partitioned with small conflicts between partitions. Scheduling is performed for the partitions and scheduling results are synthesized afterwards.

Routing is important in TAS flow scheduling, since schedulability could be affected by selected route. In some routing and scheduling schemes, flow scheduling is applied to a path predetermined for a flow. If scheduling fails for a given path, another path is found and used for scheduling. There have been research works which propose route selection schemes specifically. In general, paths are associated with a metric, sorted in regard of the metric and used in sequence for flow scheduling. In [13], each path is assigned a metric which is defined by the hop number of the path and the number of flows allocated to the path. Similarly, [22] selects a path based on the hop number and utilization level. Reference [30] selects a path based on QoS indicator including hop number of the path.

Routing and scheduling could be considered jointly not separately [23], [29], [31]. In this approach, routes with feasible slots are searched and selected when scheduling is successful. Thus feasible paths could be selected instead of optimal paths such as shortest paths. This approach is meaningful in static offline scheduling where flows to be scheduled are fixedly given. In online scheduling situation, using nonoptimal but feasible paths could improve schedulability of a flow currently being scheduled but could sacrifice schedulability of flows arriving in the future, since it could use more network resources than using optimal paths.

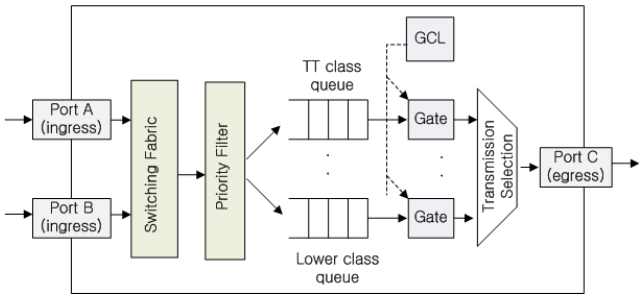


FIGURE 1. Switch structure for IEEE 802.1 Qbv time aware shaper.

In [12], joint routing and incremental scheduling under the assumption of fixed slot length and zero queuing delay is proposed. ILP problem minimizing the number of links used by a flow is formulated while scheduling assigns a slot to a flow. Since it is assumed that slot length is fixed regardless of frame length of a flow and a slot is used by at most one flow, there could be a waste of slot resource.

Multipath routing for best-effort traffic has been studied widely. In Ethernet, SPB is used in IEEE 802.1 aq [14]. In SPB, multiple equal cost paths could be used for delivery of best-effort traffic for load-balancing of a network. In [15], load balancing in SPB is studied and in [16], analysis of multipath routing is provided. However, multipath routing has been rarely studied in TAS environment. In [27], disjoint multipath routing and ILP-based scheduling scheme is proposed. However, as in [12], slot length is assumed to be fixed and a slot is reserved to only one flow, thus leading to resource waste.

III. SYSTEM MODEL

We model a network as a directed graph $G = (V, E)$ which is composed of a set of nodes V and a set of links E interconnecting the nodes. A link connecting two adjacent nodes p and q in V is represented as (p, q) where p is the transmitting node and q is the receiving node of the link. Link information is expressed in adjacency matrix $\{a(i, j)|i, j \in V\}$, where $a(i, j) = 1$ if $(i, j) \in E$. Transmission rates of the links are assumed to be equal and denoted by R.

Nodes can be Ethernet switches or Ethernet endpoints. In general, Ethernet endpoints generate frames to transmit. For simplicity, we assume that only Ethernet switches exist and data generation functions of Ethernet endpoints are included in switches. Thus the terms node and switch are used interchangeably in this work. Degree of a node denotes the number of adjacent nodes directly connected to the node by a link.

A connection for delivery of periodic time-critical TT frames from a source node to a destination node is called a flow. A pair of nodes could have several TT flows in each direction. Frames of TT flows have highest priority over other flow types and are not interfered by low-priority frames. Thus we consider only TT flows in this work. TT flow with index k is represented as f_k. Source node of f_k is denoted by s_k and destination node of f_k is denoted by t_k.

TABLE 1. The symbol table for system model.

Symbol	Description
$a(i,j)$	adjacency information of node i and j
$h(i,j)$	minimum hop count between node i and j
R	link transmission rate
f_k	TT flow k
s_k	source node of f_k
t_k	destination node of f_k
h_k	hop count of shortest paths for f_k
$r(k,i)$	i-th shortest path of f_k
$e_{k,i,h}$	h-th hop link of $r(k,i)$
$n_{k,i,h}$	transmitting node of $e_{k,i,h}$
$r(k,i,h)$	subpath of $r(k,i)$ from s_k to $n_{k,i,h}$
p_k^f	period of f_k
b_k^f	size of a frame of f_k in bits
d_k^f	transmission duration of a frame of f_k
p^H	hyperperiod length
$f_{k,u}$	u-th subflow of f_k
n_k^s	number of subflows of f_k in a hyperperiod
J_k^f	jitter bound of f_k
$J_k^{f,r}$	jitter bound ratio of f_k
T_k	a tuple of traffic characteristics of f_k
$V_{k,i,h}$	set of feasible free time slices for f_k in $e_{k,i,h}$
$v_{k,i,h,c}$	c-th free time slice in $V_{k,i,h}$
$t_{k,i,h,c}^{v,s}$	start time offset of $v_{k,i,h,c}$
$t_{k,i,h,c}^{v,e}$	end time offset of $v_{k,i,h,c}$
D_k^s	switching delay of f_k
$D(.)$	function which returns duration of a given time-slice

Fig. 1 shows the structure of a switch with TAS functionality. A frame arriving at an ingress port of a switch is forwarded to the buffer of an egress port of the switch according to its forwarding information and the priority of the frame. Each queue of an egress port is associated with a gate whose on-off operation is specified in GCL. When a time-slice is allocated to a TT flow in an outgoing link, the corresponding gate should be open during the allocated time duration. Thus gate on-off operation of a switch for TT flow is determined according to the time-slice scheduling results of the associated link.

For scheduling and configuration of TAS network, software defined network (SDN) and centralized network configuration (CNC) structure can be used [32], [33], [34]. CNC has complete knowledge of network topology and traffic characteristics of flows which are reported by end switches. CNC performs flow scheduling and configures TSN elements using network management protocols such as Netconf [35]. For TAS functionality in TSN, all nodes are time-synchronized to a master clock by IEEE 802.1 AS [36].

Each flow has associated shortest paths. We use minimum hop path as shortest path. The hop count of a path is the number of links comprising the path. Each node pair has the associated minimum hop count information, $h(i,j)$, where $i,j \in V$. h_k represents the hop count of a shortest path for flow f_k. Thus $h_k = h(s_k, t_k)$. There can be several shortest paths for a flow and i-th shortest path of f_k is denoted by $r(k,i)$. $r(k,i)$ can be expressed as

$$r(k,i) =< n_{k,i,0}, n_{k,i,1}, \ldots, n_{k,i,h_k} > \quad (1)$$

where $n_{k,i,q}$ is the q-th node in $r(k,i)$ from source node s_k, $n_{k,i,0} = s_k$ and $n_{k,i,h_k} = t_k$. Alternatively, $r(k,i)$ can be represented using links as

$$r(k,i) =< e_{k,i,0}, e_{k,i,1}, \ldots, e_{k,i,h_k-1} > \quad (2)$$

where $e_{k,i,h} = (n_{k,i,h}, n_{k,i,h+1})$. Subpath of a path is a series of links from s_k to a node on the path. $r(k,i,h)$ represents the subpath from s_k to $n_{k,i,h}$ on path $r(k,i)$. That is, $r(k,i,h)$ includes links from $e_{k,i,0}$ to $e_{k,i,h-1}$. Thus $r(k,i,h)$ can be represented as

$$r(k,i,h) =< n_{k,i,0}, n_{k,i,1}, \ldots, n_{k,i,h} > \quad (3)$$

or

$$r(k,i,h) =< e_{k,i,0}, e_{k,i,1}, \ldots, e_{k,i,h-1} > . \quad (4)$$

Note that

$$r(k,i,h_k) = r(k,i). \quad (5)$$

It is assumed that flow f_k generates equal-size frames periodically with period p_k^f and size in bits b_k^f. Transmission duration of a frame of f_k is denoted by d_k^f which is obtained as

$$d_k^f = b_k^f / R. \quad (6)$$

In real situations, a flow may generate several frames at a time. For simplicity we assume that a flow generates a frame at a time. Flows could have different periods and the least common multiple of periods of flows is called hyperperiod. Hyperperiod length is denoted by p^H.

Each delivery of a frame of a TT flow in a hyperperiod is called subflow. There are n_k^s subflows for flow f_k in a hyperperiod, where

$$n_k^s = p^H / p_k^f. \quad (7)$$

Each subflow of f_k in a hyperperiod is denoted by $f_{k,u}$, where $u = 0, 1, \ldots, n_k^s - 1$. Flow f_k has jitter bound J_k^f which is the maximum allowable jitter and jitter bound ratio $J_k^{f,r}$ which is defined as

$$J_k^{f,r} = J_k^f / p_k^f. \quad (8)$$

Traffic characteristics of flow f_k are denoted by T_k, which is a tuple of $(s_k, t_k, p_k^f, b_k^f, J_k^f)$. In no-wait switching assumed in this work, queuing delay does not exist and minimum latency could be achieved. Thus latency constraint is not considered explicitly.

TAS flow scheduling is performed on a hyperperiod basis. A hyperperiod of a link consists of time-slices with variable lengths allocated to TT flows and free time-slices not allocated yet. The length of a time-slice is measured by basic time unit with predefined granularity. It is assumed that hyperperiods of links of a network are equally aligned in time by time-synchronization. Scheduling results of a hyperperiod are repeated in other hyperperiods.

$V_{k,i,h}$ denotes the set of feasible free time-slices in a hyperperiod of $e_{k,i,h}$. $v_{k,i,h,c}$ denotes the c-th free time-slice in

$V_{k,i,h}$. $t^{v,s}_{k,i,h,c}$ denotes the start time offset from the start of a hyperperiod to the start of $v_{k,i,h,c}$ and $t^{v,e}_{k,i,h,c}$ denotes the end time offset from the start of a hyperperiod to the end of $v_{k,i,h,c}$. Duration of $v_{k,i,h,c}$ is denoted by $D(v_{k,i,h,c})$ which is obtained as

$$D(v_{k,i,h,c}) = t^{v,e}_{k,i,h,c} - t^{v,s}_{k,i,h,c} + 1 \tag{9}$$

in terms of basic time unit. Duration of $v_{k,i,h,c}$ should be no smaller than d^f_k and free time-slices smaller than d^f_k are not included in $V_{k,i,h}$. After scheduling of flow f_{k-1}, information on free time-slices of each link is updated and used for scheduling of next flow f_k. $V_{k,i,h}$ is actually a set of free time-slices in which free time-slices infeasible for f_k are removed.

In store-and-forward switching, transmission start time of a frame in an outgoing link from a switch is delayed at least by switching delay from that of an incoming link. This switching delay of flow f_k is denoted by D^s_k which is given by

$$D^s_k = d^f_k + \delta \tag{10}$$

where δ represents switch processing time. D^s_k is the minimum switching delay which could be experienced when queuing delay due to other frames does not exist. Propagation time could be included in δ. This work assumes no-wait switching by which each frame transmission of a flow in a link is delayed by D^s_k from the previous link.

IV. INCREMENTAL SINGLE PATH ROUTING AND SCHEDULING WITH NONZERO JITTER BOUND (ISPRS-NZJ)

First, we propose ISPRS-NZJ scheme, in which a shortest path is used to deliver frames of a flow. In ISPRS-NZJ, CFTSs feasible for a flow are derived in the links of a shortest path and used in scheduling. It also incorporates nonzero jitter bound.

There could be more paths other than shortest paths for a flow. However, using nonshortest paths could consume more network resources and prevent successful scheduling of future flows. Also, as hop count of a path increases, the probability of finding feasible free time-slices in all the links of the path decreases. Thus we restrict the path search space to shortest paths only.

A. DERIVATION OF CUMULATIVE FREE TIME-SLICE (CFTS)

We define CFTS in the h-th hop link of a path as delayed intersection of a CFTS of the $(h-1)$-th hop link of the path and a free time-slice of the h-th hop link. Delayed intersection is the intersection of a time-slice in the h-th hop link and a CFTS of the $(h-1)$-th hop link delayed by switching delay D^s_k. The set of CFTSs of the h-th hop link $e_{k,i,h}$ of path $r(k,i)$ is denoted by $\bar{V}_{k,i,h}$. c-th CFTS of $\bar{V}_{k,i,h}$ is denoted by $\bar{v}_{k,i,h,c}$. $\bar{V}_{k,i,h}$ is obtained as

$$\bar{V}_{k,i,h} = \bar{V}_{k,i,h-1} \cap_{D^s_k} V_{k,i,h} \tag{11}$$

where $\cap_{D^s_k}$ represents the delayed intersection operation using D^s_k.

TABLE 2. The symbol table for ISPRS-NZJ scheme.

Symbol	Description
$\bar{V}_{k,i,h}$	CFTS set in $e_{k,i,h}$
$\bar{v}_{k,i,h,c}$	c-th free time slice in $\bar{V}_{k,i,h}$
$t^{\bar{v},s}_{k,i,h,c}$	start time offset of $\bar{v}_{k,i,h,c}$
$t^{\bar{v},e}_{k,i,h,c}$	end time offset of $\bar{v}_{k,i,h,c}$
$\mathrm{SP}(k,i,h)$	a shortest path from $n_{k,i-1,h}$ to t_k
$h^u_{k,i-1}$	hop count from s_k to branching node of $r(k,i-1)$ for construction of $r(k,i)$ by depth-first way
$H^s_{k,h}$	scheduling hyperperiod for h-th hop links of paths of f_k
$I(k,u)$	index of allocated time-slice in $\bar{V}_{k,i,h_k-1}$ for $f_{k,u}$
$I_M(k,0)$	largest index of time-slice in $\bar{V}_{k,i,h_k-1}$ which can be allocated to $f_{k,0}$
$I(k,u\|I(k,0))$	index of time-slice in $\bar{V}_{k,i,h_k-1}$ allocated to $f_{k,u}$ conditioned with $I(k,0)$
$m_{k,u}$	delay margin of time-slice allocated to $f_{k,u}$
$l_{k,u}$	lacking time for time-slice allocation to $f_{k,u}$
$t^{f,s}_{k,h,u}$	transmission start time offset of $f_{k,u}$ in $e_{k,i,h}$
$t^{f,e}_{k,h,u}$	transmission end time offset of $f_{k,u}$ in $e_{k,i,h}$
$R^s_{k,u}$	feasible range of transmission start time offset of $f_{k,u}$ in the last hop link
$t^{R,s}_{k,u}$	the earliest boundary time offset of $R^s_{k,u}$
$t^{R,e}_{k,u}$	the last boundary time offset of $R^s_{k,u}$
$A_{k,h,u}$	scheduling result of $f_{k,u}$ in $e_{k,h}$ in ISPRS-NZJ

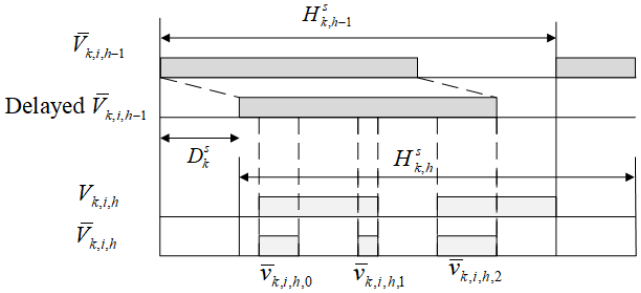


FIGURE 2. CFTS set in a link is produced by intersection of delayed CFTS set of the previous hop link and free time-slice set of the considered link.

CFTS $\bar{v}_{k,i,h,c}$ has start time offset $t^{\bar{v},s}_{k,i,h,c}$, and end time offset $t^{\bar{v},e}_{k,i,h,c}$ in a hyperperiod. If delayed intersection of CFTS $\bar{v}_{k,i,h-1,c}$ and free time-slice $v_{k,i,h,d}$ results in CFTS $\bar{v}_{k,i,h,f}$ in h-th hop link, start time offset of $\bar{v}_{k,i,h,f}$ is determined as

$$t^{\bar{v},s}_{k,i,h,f} = \max(t^{\bar{v},s}_{k,i,h-1,c} + D^s_k, t^{v,s}_{k,i,h,d}) \tag{12}$$

and end time offset is

$$t^{\bar{v},e}_{k,i,h,f} = \min(t^{\bar{v},e}_{k,i,h-1,c} + D^s_k, t^{v,e}_{k,i,h,d}). \tag{13}$$

Fig. 2 shows an example of the generation of CFTS. Note that a CFTS of $\bar{V}_{k,i,h}$ is generally shorter than the corresponding CFTS of $\bar{V}_{k,i,h-1}$ and the corresponding free time slice of $V_{k,i,h}$ due to intersection operation. In the first hop link $e_{k,i,0}$, we have

$$\bar{V}_{k,i,0} = V_{k,i,0}. \tag{14}$$

Duration of $\bar{v}_{k,i,h,c}$ is denoted by $D(\bar{v}_{k,i,h,c})$ and it is obtained as

$$D(\bar{v}_{k,i,h,c}) = t^{\bar{v},e}_{k,i,h-1,c} - t^{\bar{v},s}_{k,i,h,c} + 1 \tag{15}$$

in basic time unit. A CFTS should be feasible for the considered flow f_k, i.e.,

$$D(\bar{v}_{k,i,h-1,c}) \geq d_k^f. \tag{16}$$

Thus if the delayed intersection of two time-slices is not long enough for f_k, it is not included in $\bar{V}_{k,i,h}$. CFTS $\bar{v}_{k,i,h_k-1,c}$ in the last hop link e_{k,i,h_k-1} of flow f_k, means that there exist a series of free time-slices in the links from the source through the destination in which a frame could be transmitted with no-wait switching. Thus $\bar{v}_{k,i,h_k-1,c}$ could be allocated to f_k and scheduling for f_k could be carried out based on $\bar{V}_{k,i,h_k-1}$ of the last hop link.

Algorithm 1 ISPRS-NZJ Algorithm

Input: T_k, $\{a(i,j)\}$, $\{h(i,j)\}$
Output: $\{A_{k,h,u}\}$

1: Find $r(k,0)$
2: $i \leftarrow 0$
3: **while** 1 **do**
4: **if** $i > 0$ **then**
5: Find $h^u_{k,i-1}$ of $r(k,i-1)$
6: **if** $h^u_{k,i-1}$ is not found **then**
7: **return** Fail
8: **end if**
9: Find $r(k,i)$ from $h^u_{k,i-1}$
10: **end if**
11: $h_{init} \leftarrow 0$ if $i = 0$, $h_{init} \leftarrow h^u_{k,i-1}$ if $i > 0$
12: **for** $h = h_{init}$ to $h_k - 1$ **do**
13: Find CFTS set $\bar{V}_{k,i,h}$
14: **end for**
15: $\{A_{k,h_k-1,u}\} \leftarrow \text{Sch}(\bar{V}_{k,i,h_k-1})$ (Alg. 2)
16: **if** $\{A_{k,h_k-1,u}\}$ is found **then**
17: **for** $h = h_k - 2$ to 0 **do**
18: **for** $u = 0$ to $n^s_k - 1$ **do**
19: Find scheduling result $A_{k,h,u}$ of $f_{k,u}$
20: **end for**
21: **end for**
22: Update free time-slice information in the links
23: **return** Success
24: **else**
25: $i \leftarrow i + 1$
26: **end if**
27: **end while**

B. SINGLE PATH ROUTING AND SCHEDULING WITH NONZERO JITTER BOUND ALGORITHM

Alg. 1 shows the proposed ISPRS-NZJ algorithm for flow f_k. A shortest path used for scheduling is obtained one by one. The algorithm starts with finding a first shortest path for the flow (line 1). If flow scheduling fails for path $r(k, i-1)$, another shortest path $r(k,i)$ is found and examined for scheduling.

Depth-first approach is used for consecutive path finding as follows. Node $n_{k,i-1,h}$ is the node in $r(k,i-1)$ which is h hop away from source s_k and has minimum hop count $h_k - h$ to t_k. $n_{k,i-1,0}$ is source s_k and $n_{k,i-1,h_k}$ is destination t_k. $n_{k,i-1,h}$ could have unvisited neighbor nodes which have shortest paths to t_k. Among $\{n_{k,i-1,h|h=0,1,\ldots,h_k-2}\}$, the node with unvisited neighbor nodes having shortest paths to t_k and closest to t_k is selected for the next path. The hop count from s_k to the selected node is denoted by $h^u_{k,i-1}$ (line 5). Thus $n_{k,i-1,h^u_{k,i-1}}$ is the branching node of path $r(k,i-1)$ for next path $r(k,i)$. Links after branching node are changed by path update as shown in Fig. 3. With $h^u_{k,i-1}$, $r(k,i)$ is obtained as

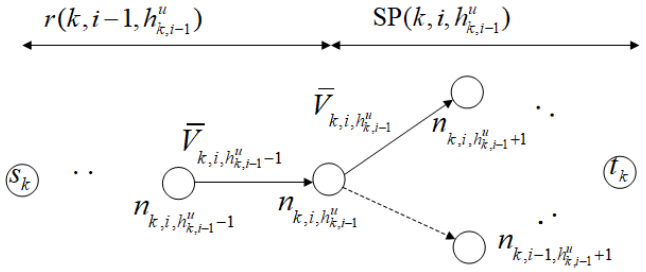


FIGURE 3. Links after branching node are changed by path update. CFTS sets of the newly included links are updated in ISPRS-NZJ.

$$r(k,i) =< r(k,i-1,h^u_{k,i-1}), \text{SP}(k,i,h^u_{k,i-1}) > \tag{17}$$

where $\text{SP}(k,i,h^u_{k,i-1})$ is a shortest path from $n_{k,i-1,h^u_{k,i-1}}$ to t_k included in $r(k,i)$ (line 9). Then visit information of nodes in $\text{SP}(k,i,h^u_{k,i-1})$ are reset in $r(k,i)$. The possible smallest value of $h^u_{k,i-1}$ is 0, which corresponds to source s_k being the branching node. If all the shortest paths are examined without scheduling success, $h^u_{k,i-1}$ is not found and scheduling for flow f_k fails (line 6 and 7).

Once $r(k,i)$ is found, CFTS sets in the links of $r(k,i)$ are derived. In the first path $r(k,0)$, CFTS sets should be derived in all the links of the path (line 11 - 13). In paths other than $r(k,0)$, subpath $r(k,i-1,h^u_{k,i-1})$ is common in $r(k,i-1)$ and $r(k,i)$. Thus CFTS sets in subpath $r(k,i-1,h^u_{k,i-1})$ in $r(k,i-1)$ are reused in $r(k,i)$ and we have

$$\bar{V}_{k,i,q} = \bar{V}_{k,i-1,q}, \quad q = 0,1,\ldots,h^u_{k,i-1} - 1. \tag{18}$$

CFTSs in the links after $n_{k,i-1,h^u_{k,i-1}}$ in $r(k,i)$ are obtained as described in (11) and (14) and shown in Fig. 3 (line 12 and 13).

Based on the CFTS set in the last hop link $\bar{V}_{k,i,h_k-1}$, allocation of time-slices to subflows of f_k is performed by scheduling function Sch(·) (line 15). Details of Sch(·) are explained in Alg. 2.

If time-slice allocation is successful, scheduling results $\{A_{k,h,u}\}$ of subflows in all the links comprising the path is found (line 17-21). $A_{k,h,u}$ consists of transmission start time offset of $f_{k,u}$ in $e_{k,h}$ and the transmitting node of the link. When $r(k,i)$ is used, $A_{k,h,u}$ is defined as

$$A_{k,h,u} = (t^{f,s}_{k,h,u}, n_{k,i,h}) \tag{19}$$

where $t^{f,s}_{k,h,u}$ is the frame transmission start time offset of $f_{k,u}$ in $e_{k,h}$ of $r(k,i)$. $t^{f,s}_{k,h_k-1,u}$ in the last hop link is obtained by

Alg. 2. Then $t^{f,s}_{k,h,u}$ in other links are obtained as (line 19)

$$t^{f,s}_{k,h,u} = t^{f,s}_{k,h_k-1,u} - (h_k - 1 - h)D^s_k, \quad h = 0, 1, \ldots, h_k - 2 \tag{20}$$

under no-wait switching assumption.

With scheduling success, information on free time-slices in the scheduled links are updated according to the scheduling result (line 22) and scheduling ends successfully (line 23). A free time-slice is divided into maximum two free time-slices by allocation to a subflow. Thus the number of free time-slices in a link has $O(N)$ relation with N scheduled flows. Meanwhile, if length of a resulting free time-slice is below the required duration of minimum size frame, the time-slice is removed from the updated set of free time-slices. When the number of scheduled flows is large, duration of a free time-slice tends to be small. In this case, a time-slice allocated to a subflow is less likely to be divided into two free time-slices. Thus the increase rate of the number of free time-slices in a link would diminish as the number of scheduled flows increases.

If time-slice allocation fails for path $r(k, i)$, i is increased by 1 (line 25) and search for the next path starts again (line 4 and 5).

C. TIME-SLICE ALLOCATION

With fixed switching delay, the earliest possible start time offset of a CFTS in a default hyperperiod in $e_{k,i,h}$ of $r(k, i)$ is hD^s_k. Thus we define scheduling hyperperiod for f_k in $e_{k,i,h}$ as the duration which starts at hD^s_k and ends at $p^H + hD^s_k - 1$. If the end time offset of a scheduling hyperperiod is larger than p^H, wrap-around is used, i.e., early part of a default hyperperiod becomes the latter part of a scheduling hyperperiod. Scheduling hyperperiod for $e_{k,i,h}$ is denoted by $H^s_{k,h}$. Fig. 2 shows an example of scheduling hyperperiod delayed by D^s_k from the scheduling hyperperiod of the previous hop link.

Time-slice allocation for flow f_k in path $r(k, i)$ is performed by time-slice allocation for each subflow $f_{k,u}$ based on CFTS set $\bar{V}_{k,i,h_k-1}$ in the last hop link of the path in scheduling hyperperiod H^s_{k,h_k-1}. As a result of successful time-slice allocation, transmission start time offset of each subflow $f_{k,u}$ in the last hop link, $t^{f,s}_{k,h_k-1,u}$ is obtained, which is used to determine transmission start time offsets in other links of the path.

$I(k, u)$ denotes the index of CFTS in $\bar{V}_{k,i,h_k-1}$ allocated to $f_{k,u}$. $I_M(k, 0)$ is the maximum index of CFTS in $\bar{V}_{k,i,h_k-1}$ which can be allocated to $f_{k,0}$. If latency bound is assumed to be larger than flow period, transmission start time offset of $f_{k,0}$ in the last hop link could be set smaller than $h_k D^s_k + p^f_k$. Thus $I_M(k, 0)$ dictates the last time-slice in $\bar{V}_{k,i,h_k-1}$ whose start time offset is smaller than $h_k D^s_k + p^f_k$.

CFTS allocations to subflows other than $f_{k,0}$ are dependent on the CFTS allocation to $f_{k,0}$. $I(k, u|I(k, 0))$ indicates the index of CFTS allocated to $f_{k,u}$ conditioned with $I(k, 0)$. We allocate CFTS with index from 0 through $I_M(k, 0)$ to $f_{k,0}$ one by one and check whether allocations to other subflows are possible conditioned with each $I(k, 0)$. There can be several successful CFTS allocation sets for the subflows for each $I(k, 0)$. Then one of the CFTS allocation sets is selected with specific $I(k, 0)$ and used. There could be various criterions for selecting $I(k, 0)$. We use a simple criterion which is to select the smallest $I(k, 0)$. Thus if a set of CFTS allocations associated with $I(k, 0)$ is found first, no more search for CFTS allocation is performed.

Alg. 2 shows the time-slice allocation algorithm for a flow using CFTS. At first, $I_M(k, 0)$ is obtained as explained above (line 1). Then CFTS $\bar{v}_{k,i,h_k-1,c}$ is temporarily allocated to $f_{k,0}$ with c for $I(k, 0)$ starting from 0 (line 2 and 3) and the start time offset of the CFTS, $t^{\bar{v},s}_{k,i,h_k-1,c}$ is correspondingly set as the transmission start time offset of $f_{k,0}$ in the last hop link (line 4).

Duration of a CFTS is no smaller than frame transmission duration d^f_k. Thus when a CFTS is allocated to a subflow,

Algorithm 2 Time-Slice Allocation Function Sch(·)

Input: T_k, $\bar{V}_{k,i,h_k-1}$
Output: $\{A_{k,h_k-1,u}\}$

1: Find $I_M(k, 0)$ in $\bar{V}_{k,i,h_k-1}$
2: **for** $c = 0$ to $I_M(k, 0)$ **do**
3: Allocate $\bar{v}_{k,i,h_k-1,c}$ to $f_{k,0}$
4: $t^{f,s}_{k,h_k-1,0} \leftarrow t^{\bar{v},s}_{k,i,h_k-1,c}$
5: Calculate $m_{k,0}$
6: **for** $u = 1$ to $n^s_k - 1$ **do**
7: Find $R^s_{k,u}$ for $f_{k,u}$
8: Allocate CFTS to $f_{k,u}$ and set $t^{f,s}_{k,h_k-1,u}$
9: **if** Allocation succeeds for $f_{k,u}$ **then**
10: Calculate $m_{k,u}$
11: **else**
12: Calculate lacking time $l_{k,u}$
13: **if** $l_{k,u} \le \min(\{m_{k,q} | q = 0, \ldots, u-1\})$ **then**
14: Allocate CFTS to $f_{k,u}$ and set $t^{f,s}_{k,h_k-1,u}$
15: Calculate $m_{k,u}$
16: **for** $q = 0$ to $u - 1$ **do**
17: $t^{f,s}_{k,h_k-1,q} \leftarrow t^{f,s}_{k,h_k-1,q} + l_{k,u}$
18: $m_{k,q} \leftarrow m_{k,q} - l_{k,u}$
19: **end for**
20: **else if** $l_{k,u} \le m_{k,0}$ **then**
21: $t^{f,s}_{k,h_k-1,0} \leftarrow t^{f,s}_{k,h_k-1,0} + l_{k,u}$
22: $m_{k,0} \leftarrow m_{k,0} - l_{k,u}$
23: $u \leftarrow 1$
24: **else**
25: $u \leftarrow n^s_k$
26: **end if**
27: **end if**
28: **end for**
29: **end for**
30: **if** $t^{f,s}_{k,h_k-1,0} \ge h_k D^s_k + p^f_k$ **then**
31: **return** Fail
32: **end if**

various transmission start times within the CFTS are possible. We define delay margin $m_{k,u}$ of $f_{k,u}$ as the amount of time that the transmission can be delayed in the currently assigned CFTS given current transmission start time offset in the CFTS. This is the remaining time in the currently assigned CFTS after the transmission end of the frame. Thus $m_{k,u}$ is obtained as

$$m_{k,u} = t^{\bar{v},e}_{k,i,h_k-1,g} - t^{f,e}_{k,h_k-1,u} \tag{21}$$

where $t^{f,e}_{k,h_k-1,u}$ is the transmission end time offset of $f_{k,u}$ in the last hop link and g is the index of CFTS allocated to $f_{k,u}$ conditioned with $I(k,0)$, i.e., $g = I(k,u|I(k,0))$. $t^{f,e}_{k,h_k-1,u}$ is given by

$$t^{f,e}_{k,h_k-1,u} = t^{f,s}_{k,h_k-1,u} + d^f_k - 1 \tag{22}$$

in basic time unit. When a CFTS is allocated to $f_{k,u}$ and transmission start time offset in the CFTS is set, corresponding $m_{k,u}$ is calculated (line 5, 10 and 15).

$R^s_{k,u}$ denotes the feasible range of transmission start time offset of subflow $f_{k,u}$ other than $f_{k,0}$ in the last hop link. In time-slice allocation, $t^{f,s}_{k,h_k-1,u}$ should be within $R^s_{k,u}$, i.e.,

$$t^{R,s}_{k,u} \le t^{f,s}_{k,h_k-1,u} \le t^{R,e}_{k,u} \tag{23}$$

where $t^{R,s}_{k,u}$ denotes the earliest boundary time offset of $R^s_{k,u}$ and $t^{R,e}_{k,u}$ denotes the last boundary time offset. $R^s_{k,u}$ is determined based on $t^{f,s}_{k,h_k-1,0}$ and jitter bound J^f_k (line 7). Thus $t^{R,s}_{k,u}$ is obtained as the sum of $t^{f,s}_{k,h_k-1,0}$ and u-times of flow period p^f_k as follows:

$$t^{R,s}_{k,u} = t^{f,s}_{k,h_k-1,0} + up^f_k. \tag{24}$$

And $t^{R,e}_{k,u}$ is obtained as the sum of $t^{R,s}_{k,u}$ and J^f_k as follows:

$$t^{R,e}_{k,u} = t^{R,s}_{k,u} + J^f_k. \tag{25}$$

Given $R^s_{k,u}$, a CFTS capable of serving $f_{k,u}$ is found in $\bar{V}_{k,i,h_k-1}$ by linear search and $t^{f,s}_{k,h_k-1,u}$ is set accordingly as follows (line 8). If start time offset of CFTS $\bar{v}_{k,i,h_k-1,g}$ is within $R^s_{k,u}$, this CFTS can be allocated to $f_{k,u}$ and $t^{f,s}_{k,h_k-1,u}$ is set to the start time offset of the CFTS, i.e.,

$$t^{f,s}_{k,h_k-1,u} = t^{\bar{v},s}_{k,i,h_k-1,g} \tag{26}$$

if

$$t^{R,s}_{k,u} \le t^{\bar{v},s}_{k,i,h_k-1,g} \le t^{R,e}_{k,u}. \tag{27}$$

If start time offset of a CFTS is earlier than $t^{R,s}_{k,u}$, i.e., $t^{\bar{v},s}_{k,i,h_k-1,g} < t^{R,s}_{k,u}$ and the remaining time in the CFTS after $t^{R,s}_{k,u}$ is long enough for the flow, this CFTS can be allocated to $f_{k,u}$. In this case,

$$t^{f,s}_{k,h_k-1,u} = t^{R,s}_{k,u} \tag{28}$$

if

$$t^{R,s}_{k,u} + d^f_k - 1 \le t^{\bar{v},e}_{k,i,h_k-1,g}. \tag{29}$$

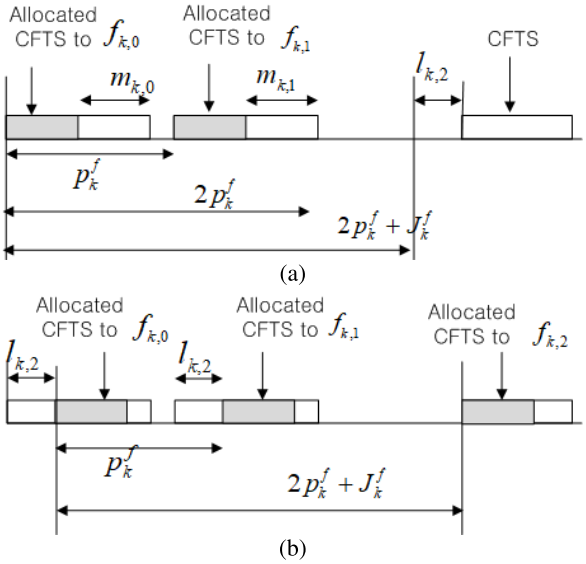


FIGURE 4. Flow scheduling using delay margin. Initially subflow $f_{k,2}$ cannot be allocated CFTS due to lacking time $l_{k,2}$. By delaying allocated transmission times of $f_{k,0}$ and $f_{k,1}$ by $l_{k,2}$, $f_{k,2}$ can be allocated a time-slice.

Once time-slice allocation is successful, delay margin $m_{k,u}$ is obtained by (21) (line 10). Note that transmission start time offsets are not fixedly determined but can be changed during scheduling for other subflows of the flow.

Assume that we have allocated CFTSs and set transmission start time offsets of subflows $\{f_{k,q}|q = 0, .., u-1\}$. If there is no feasible CFTS for the next subflow $f_{k,u}$ to be scheduled, delaying the transmission start time offsets of currently scheduled subflows could enable allocation success for $f_{k,u}$. Consider the case that two consecutive CFTSs with indices g and $g+1$ have a relation such that $t^{\bar{v},e}_{k,i,h_k-1,g} < t^{R,s}_{k,u} + d^f_k - 1$ and $t^{\bar{v},s}_{k,i,h_k-1,g+1} > t^{R,e}_{k,u}$. In this case, CFTS $\bar{v}_{k,i,h_k-1,g}$ is too early to serve subflow $f_{k,u}$ and the next CFTS $\bar{v}_{k,i,h_k-1,g+1}$ is too late. Thus $\bar{v}_{k,i,h_k-1,g+1}$ has lacking time $l_{k,u}$ to be allocated to $f_{k,u}$ which is given by (line 12)

$$l_{k,u} = t^{\bar{v},s}_{k,i,h_k-1,g+1} - t^{R,e}_{k,u}. \tag{30}$$

Lacking time 0 means that CFTS allocation to the subflow is successful.

If $l_{k,u}$ is larger than 0 but smaller than the minimum of delay margins $\{m_{k,q}|q = 0, 1, \ldots, u-1\}$ of the scheduled subflows, transmission times of the scheduled subflows could be delayed by $l_{k,u}$ for scheduling of $f_{k,u}$ while maintaining the current assignment of CFTSs to them. That is, if $0 < l_{k,u} \le \min(\{m_{k,q}|q = 0, \ldots, u-1\})$, $t^{f,s}_{k,h_k-1,u}$ is determined as (line 13 and 14)

$$t^{f,s}_{k,h_k-1,u} = t^{\bar{v},s}_{k,i,h_k-1,g+1}. \tag{31}$$

And delay margin of $f_{k,u}$ is determined by (21) with $g+1 = I(k,u|I(k,0))$ (line 15). Moreover $t^{f,s}_{k,h_k-1,q}$ and $m_{k,q}$ of the subflows scheduled earlier, $\{f_{k,q}|q = 0, .., u-1\}$ are updated accordingly (line 17 and 18).

If $l_{k,u}$ is larger than minimum of delay margins $\{m_{k,q}|q = 0, 1, \ldots, u-1\}$ and no larger than $m_{k,0}$ (line 20), the CFTS

TABLE 3. The symbol table for IMPRS-NZJ scheme.

Symbol	Description
$P^m_{k,i}$	set of paths $\{r(k,j)\|j=0,1,..,i\}$ used in IMPRS-NZJ
$E^m_{k,i,h}$	set of h-th hop links in $P^m_{k,i}$
$e^m_{k,i,h,d}$	d-th link in $E^m_{k,i,h}$
$N(E^m_{k,i,h})$	size of $E^m_{k,i,h}$
$N^m_{k,i,h}$	set of h-th hop nodes in $P^m_{k,i}$
$N(N^m_{k,i,h})$	size of $N^m_{k,i,h}$
$n^m_{k,i,h,c}$	c-th node in $N^m_{k,i,h}$
$V^m_{k,i,h,d}$	set of feasible free time-slices in $e^m_{k,i,h,d}$
$\bar{V}^m_{k,i,h,d}$	set of CFTSs in $e^m_{k,i,h,d}$
$\bar{v}^m_{k,i,h,d,f}$	f-th free time slice in $\bar{V}^m_{k,i,h,d}$
$\hat{V}^m_{k,i,h,c}$	set of CCFTSs at $n^m_{k,i,h,c}$
$\hat{v}^m_{k,i,h,c,f}$	f-th CCFTS in $\hat{V}^m_{k,i,h,c}$
$n^{m,p}_{k,i,h,c,f}$	source node of the incoming link having $\hat{v}^m_{k,i,h,c,f}$ to $n^m_{k,i,h,c}$
$A^m_{k,h,u}$	scheduling result of IMPRS-NZJ for $f_{k,u}$ in h-th hop links
$n^{m,t}_{k,h,u}$	transmitting node of h-th hop link used by $f_{k,u}$ in IMPRS-NZJ

allocated to $f_{k,0}$ is maintained with transmission start time offset of $f_{k,0}$ and $m_{k,0}$ adjusted by $l_{k,u}$ (line 21 and 22). Subflows from $f_{k,1}$ are rescheduled conditioned with the new $t^{f,s}_{k,h_k-1,0}$ (line 22).

If $l_{k,u} > m_{k,0}$, delayed allocation cannot be applied. Thus the scheduling of the flow with current $I(k,0)$ ends (line 25) and the CFTS next to the current $I(k,0)$ is newly allocated to $f_{k,0}$ and scheduling restarts for the subflows $\{f_{k,u}|u = 1,2,\ldots n^s_k - 1\}$ (line 2).

We restrict $t^{f,s}_{k,h_k-1,0}$ to be smaller than $h_k D^s_k + p^f_k$. If $t^{f,s}_{k,h_k-1,0}$ within this range cannot be found, it is decided as scheduling failure (line 30 and 31).

Fig. 4a and 4b show an example of flow scheduling using delay margin. In Fig. 4a, subflow $f_{k,2}$ cannot be allocated due to nonzero lacking time $l_{k,2}$. This situation is solved in Fig. 4b by delaying the transmissions of $f_{k,0}$ and $f_{k,1}$ by $l_{k,2}$ under the constraint of nonzero jitter bound.

In this way, scheduling in the last hop link produces a set of CFTSs allocated to the subflows of a flow corresponding to an assignment of $I(k,0)$ while satisfying jitter bound. This scheduling result is used to set the transmission start time offsets in other links of the path in Alg. 1.

Operation of ISPRS-NZJ scheme consists of derivation of CFTS sets, time-slice allocation based on CFTS set of the last hop link and update of free time-slice information of scheduled links according to scheduling result. All these procedures are dependent on the number of free time-slices. The number of free time-slices in a link is $O(N)$ for N scheduled flows. With $O(N)$ free time-slices, scheduling of a new flow consumes $O(N)$ operations. Thus the time complexity of ISPRS-NZJ scheme is $O(N^2)$ for N flows.

V. INCREMENTAL MULTIPATH ROUTING AND SCHEDULING WITH NONZERO JITTER BOUND (IMPRS-NZJ)

In IMPRS-NZJ scheduling, each frame of a flow is delivered in one of multiple shortest paths. Thus frames of different subflows could use different paths. Scheduling is performed based on CCFTSs of the destination node. Allocation of time slices in CCFTSs to subflows and determination of transmission start time offsets of the subflows are carried out in a similar way to ISPRS scheme, except that CCFTSs of the destination node are used instead of CFTSs of the last hop link. If flow scheduling does not succeed for a set of paths, another path is added if available and new CCFTS sets for the increased set of paths are obtained and used for scheduling.

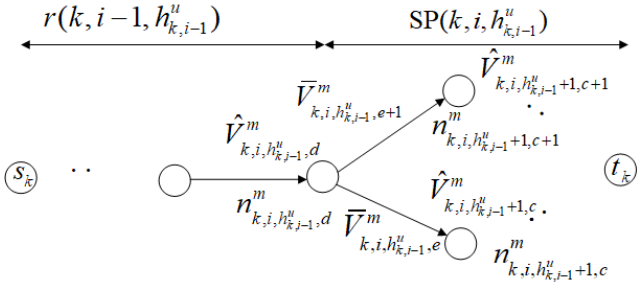


FIGURE 5. In IMPRS-NZJ scheme, each node of a path derives CCFTS based on CFTSs of the incoming links. CFTS of a link is obtained from delayed intersection of CCFTS of the transmitting node of the link and free time-slices of the link.

A. COMBINED CUMULATIVE FREE TIME-SLICE (CCFTS)

CCFTSs of a node are union of CFTSs of the incoming links to the node on the given paths of a flow. Unlike ISPRS-NZJ, there could be several h-th hop links used at the same time in IMPRS-NZJ. $P^m_{k,i}$ denotes the set of paths considered for multipath scheduling of flow f_k in which paths from $r(k,0)$ to $r(k,i)$ are included, i.e.,

$$P^m_{k,i} = \{r(k,j)|j = 0,1,..,i\}. \tag{32}$$

The set of links h hop away from source s_k in $P^m_{k,i}$ is represented as $E^m_{k,i,h}$ and a link with index d in $E^m_{k,i,h}$ is denoted by $e^m_{k,i,h,d}$. $N(E^m_{k,i,h})$ represents the size of $E^m_{k,i,h}$. $\bar{V}^m_{k,i,h,d}$ denotes the set of CFTSs in link $e^m_{k,i,h,d}$ where $d = 0,1,\ldots,N(E^m_{k,i,h})-1$. The set of nodes at h-th hop away from source s_k on $P^m_{k,i}$ are denoted by $N^m_{k,i,h}$. $N(N^m_{k,i,h})$ represents the size of $N^m_{k,i,h}$. Each node in $N^m_{k,i,h}$ is represented as $n^m_{k,i,h,c}$ where $c = 0,1,\ldots,N(N^m_{k,i,h})-1$. The set of CCFTSs at node $n^m_{k,i,h,c}$ is denoted by $\hat{V}^m_{k,i,h,c}$.

If scheduling of flow f_k fails with $P^m_{k,i-1}$, another shortest path $r(k,i)$ is found as in (17) and added resulting in $P^m_{k,i}$. Then scheduling is performed again with $P^m_{k,i}$. That is,

$$P^m_{k,i} = P^m_{k,i-1} \cup r(k,i). \tag{33}$$

Fig. 5 shows the path update and derivation of associated CCFTS set and CFTS set. CCFTSs of the nodes from s_k through $n^m_{k,i-1,h^u_{k,i-1}}$ are reused in $P^m_{k,i}$. Thus we have

$$\hat{V}^m_{k,i,h,p} = \hat{V}^m_{k,i-1,h,p}, \quad h = 1,\ldots,h^u_{k,i-1} \tag{34}$$

where $n_{k,i,h} = n^m_{k,i-1,h,p}$. Derivations of new CFTSs and CCFTSs start after $n^m_{k,i-1,h^u_{k,i-1}}$ and proceed toward destination t_k.

CFTS of a link in IMPRS-NZJ is obtained by delayed intersection of CCFTS set of the transmitting node of the link and the set of feasible free time-slices of the link. Thus the set of CFTSs in h-th hop link $e^m_{k,i,h,e}$ is obtained as

$$\bar{V}^m_{k,i,h,e} = \hat{V}^m_{k,i,h,d} \cap_{D^s_k} V^m_{k,i,h,e} \tag{35}$$

where $n^m_{k,i,h,d}$ is the transmitting node of the considered link $e^m_{k,i,h,e}$ and $V^m_{k,i,h,e}$ is the set of feasible free time-slices of the link.

For $h > h^u_{k,i-1}$, CCFTS at $n_{k,i,h}$ is obtained as union of the previous CCFTS set of $n_{k,i,h}$ and the CFTS set of link $e_{k,i,h-1}$ as

$$\hat{V}^m_{k,i,h,d} = \hat{V}^m_{k,i-1,h,d} \cup \bar{V}^m_{k,i,h-1,c} \tag{36}$$

where $n_{k,i,h} = n^m_{k,i,h,d}$ and $e_{k,i,h-1} = e^m_{k,i,h-1,c}$. If $n_{k,i,h}$ is newly added, i.e., $n_{k,i,h} \notin N^m_{k,i-1,h}$, CCFTS at $n_{k,i,h}$ is obtained as

$$\hat{V}^m_{k,i,h,d} = \bar{V}^m_{k,i,h-1,c}. \tag{37}$$

In the union operation for CCFTS, if a time-slice is enclosed entirely in another time-slice, it is removed.

At node $n^m_{k,i,1,d}$ next to source s_k, CCFTS set is equal to the set of feasible free time-slices in the corresponding first hop link $e^m_{k,i,0,c}$, i.e.,

$$\hat{V}^m_{k,i,1,d} = V^m_{k,i,0,c} \tag{38}$$

where $V^m_{k,i,0,c}$ is the set of feasible free time-slices of link $e^m_{k,i,0,c}$.

Note that CFTS for flow f_k is long enough for transmission of the frame of the flow, i.e.,

$$D(\bar{v}^m_{k,i,h,e,f}) \geq d^f_k. \tag{39}$$

Each CCFTS $\hat{v}^m_{k,i,h,d,f}$ in $\hat{V}^m_{k,i,h,d}$ has its associated previous hop node which is represented as $n^{m,p}_{k,i,h,d,f}$, which is the source node of the incoming link to node $n^m_{k,i,h,d}$ having the CCFTS $\hat{v}^m_{k,i,h,d,f}$. This information is used in time-slice allocation in the links used by f_k by *backtracking*.

B. MULTIPATH ROUTING AND SCHEDULING WITH NONZERO JITTER BOUND ALGORITHM

Alg. 3 shows the proposed IMPRS-NZJ algorithm. Since IMPRS-NZJ scheme is similar to ISPRS-NZJ, differences between two schemes are mainly presented here. As in ISPRS-NZJ, when scheduling fails with path $r(k, i-1)$, another path $r(k, i)$ is found by depth-first search using $h^u_{k,i-1}$ (line 4 - 10). For each link $e_{k,i,h}$ in SP$(k, i, h^u_{k,i-1})$, CFTS set $\bar{V}^m_{k,i,h,e}$ of the link is derived as in (35) (line 13). CCFTS set of the end node of $e_{k,i,h}$, $n^m_{k,i,h+1,d}$ is obtained based on the CFTS sets of the incoming links as in (36), (37) and (38) (line 14).

Allocation of time-slices to subflows of f_k with $P^m_{k,i}$ is done based on the CCFTS set at the destination node, $\hat{V}^m_{k,i,h_k,0}$ where $n^m_{k,i,h_k,0}$ is the destination node t_k (line 16). Time-slice allocation procedure Sch$(\cdot)$ based on CCFTS set of the destination node is almost equal to Alg. 2 based on the CFTS set of the last hop link, except that CCFTS set is used instead of CFTS set. Thus detailed explanation of Sch$(\hat{V}^m_{k,i,h_k,0})$ is omitted here. CCFTSs at the destination node are actually CFTSs in the last hop links. Thus scheduling based on $\hat{V}^m_{k,i,h_k,0}$ is done in scheduling hyperperiod H^s_{k,h_k-1}.

If time-slice allocation fails, i is increased by 1 (line 26) and search for next path is started again (line 4 and 5). If time-slice allocation to $\{f_{k,u}\}$ at the destination node is successful (line 17), allocation results $\{A^m_{k,h,u}\}$ of $f_{k,u}$ in all the links used by f_k are found (line 20). $A^m_{k,h,u}$ consists of transmission start time offset and transmitting node for each subflow $f_{k,u}$ in each h-th hop link. $A^m_{k,h,u}$ is defined as

$$A^m_{k,h,u} = (t^{f,s}_{k,h,u}, n^{m,t}_{k,h,u}) \tag{40}$$

where $t^{f,s}_{k,h,u}$ is the frame transmission start time offset of $f_{k,u}$ in h-th hop link and $n^{m,t}_{k,h,u}$ represents the transmitting node in h-th hop link. $A^m_{k,h,u}$ is obtained by *backtracking* starting from $A^m_{k,h_k-1,u}$. Given $t^{f,s}_{k,h_k-1,u}$ in the last hop link, $t^{f,s}_{k,h,u}$ is obtained by (20). $n^{m,t}_{k,h,u}$ is obtained as

$$n^{m,t}_{k,h,u} = n^{m,p}_{k,i,h+1,c,f} \tag{41}$$

Algorithm 3 IMPRS-NZJ Algorithm

Input: T_k, $\{a(i,j)\}$, $\{h(i,j)\}$
Output: $\{A^m_{k,h,u}\}$

1: Find $r(k, 0)$
2: $i \leftarrow 0$
3: **while** 1 **do**
4: **if** $i > 0$ **then**
5: Find $h^u_{k,i-1}$ of $r(k, i-1)$
6: **if** $h^u_{k,i-1}$ is not found **then**
7: **return** Fail
8: **end if**
9: Find $r(k, i)$ from $h^u_{k,i-1}$
10: **end if**
11: $h_{init} \leftarrow 0$ if $i = 0$, $h_{init} \leftarrow h^u_{k,i-1}$ if $i > 0$
12: **for** $h = h_{init}$ to $h_k - 1$ **do**
13: Find CFTS set $\bar{V}^m_{k,i,h,e}$ of $e^m_{k,i,h,e} = e_{k,i,h}$
14: Find CCFTS set $\hat{V}^m_{k,i,h+1,d}$ of $n^m_{k,i,h+1,d} = n_{k,i,h+1}$
15: **end for**
16: $\{A^m_{k,h_k-1,u}\} \leftarrow$ Sch$(\hat{V}^m_{k,i,h_k,0})$
17: **if** $\{A^m_{k,h_k-1,u}\}$ is found **then**
18: **for** $h = h_k - 2$ to 0 **do**
19: **for** $u = 0$ to $n^s_k - 1$ **do**
20: Find $A^m_{k,h,u}$ by *backtracking*
21: **end for**
22: **end for**
23: Update free time-slice information in the links
24: **return** Success
25: **else**
26: $i \leftarrow i + 1$
27: **end if**
28: **end while**

TABLE 4. The parameter table for numerical results.

Parameter	Values
Default minimum degree	7
Default number of switches	20
Default number of flows	500
Default mean frame size μ	300 bytes
Maximum frame size	1500 bytes
Minimum frame size	64 bytes
Standard deviation of frame size σ	0.5 μ
Link transmission rate R	500 Mbps
Basic time unit granularity	200 ns
Switch processing time δ	2 μs
Flow periods	200, 250, 500, 1000 μs
Hyperperiod length	1000 μs
Confidence level	95 %
Margin of error	5 %

where $n^{m,p}_{k,i,h+1,c,f}$ is the source node of $\hat{v}^{m}_{k,i,h+1,c,f}$ containing the time slice allocated to $f_{k,u}$ at the receiving node of h-th hop link. Information on free time-slices in the scheduled links is updated according to the scheduling result.

Operation of IMPRS-NZJ is similar to ISPRS-NZJ except that derivations of CCFTSs at nodes in the paths are necessary. Thus time complexity of IMPRS-NZJ for N flows is also $O(N^2)$ as in ISPRS-NZJ.

VI. NUMERICAL RESULTS

Simulation program for the proposed schemes was developed in MATLAB. For network configuration, locations of switches were assigned randomly and for each switch, other switches located close are connected with links to this switch to satisfy the minimum degree. Parameter values used in numerical evaluation are summarized in Table 4. Number of flows means the number of TT flows to be scheduled in a network. Truncated normal distribution was used for frame size with mean μ and standard deviation σ. The periods of flows were uniformly distributed among $200\mu s$, $250\mu s$, $500\mu s$, $1000\mu s$. Fig. 6 shows an example of network configuration with minimum degree of 7. Simulation runs were decided based on the margin of error and confidence level. Default parameter values were used if not stated otherwise.

The results in this section have been carried out on a computer with an Intel Core i5-9400 processor running at 2.9 GHz and 24 GB RAM.

For comparison, we also implemented time-tabling method with zero jitter in [11]. Time-tabling method performed flow scheduling for a shortest path as in ISPRS-NZJ scheme. In time-tabling method, flow scheduling results with earliest possible transmission time offset in a hyperperiod were obtained.

Fig. 7 shows that flow schedule failure rates with respect to jitter bound ratio of flows, when mean frame size μ is 1000 or 600 bytes and the number of flows is 1000. Flow schedule failure rate is defined as the ratio of the number of TT flows failed in scheduling over the number of TT flows to be scheduled. Jitter bound ratio is the ratio of jitter bound over flow period. Jitter bound ratio of 0 means that no jitter

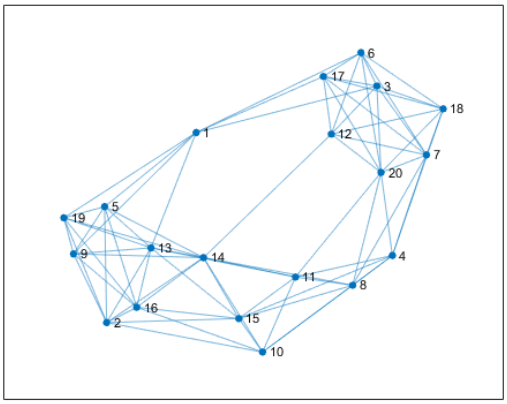


FIGURE 6. An example of network configuration with minimum degree of 7.

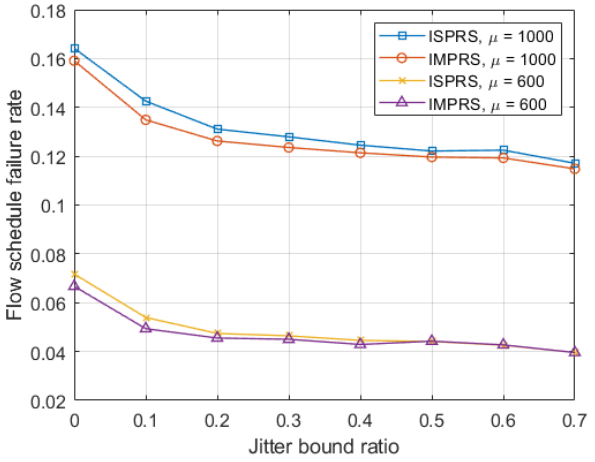


FIGURE 7. Flow schedule failure rates with respect to the jitter bound ratios.

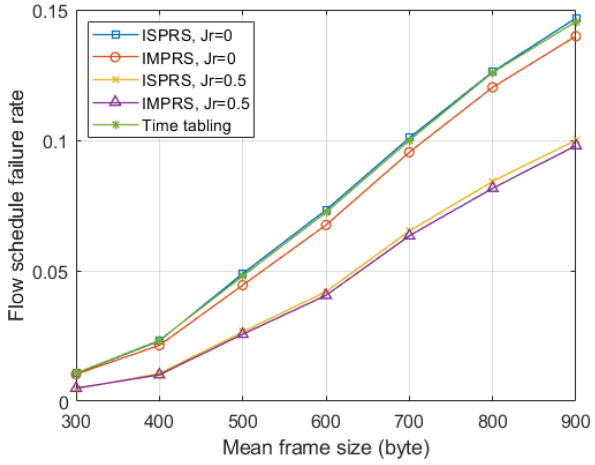


FIGURE 8. Flow schedule failure rates with respect to mean frame size.

is permitted. ISPRS and IMPRS are used to represent the proposed schemes regardless of jitter bound size. ISPRS and IMPRS with nonzero jitter bound correspond to ISPRS-NZJ and IMPRS-NZJ respectively. As jitter bound ratio increases from zero, schedule failure rates of both ISPRS-NZJ and IMPRS-NZJ schemes decrease compared with ISPRS or IMPRS with zero jitter. IMPRS-NZJ shows lower schedule failure rates than ISPRS-NZJ especially when schedule

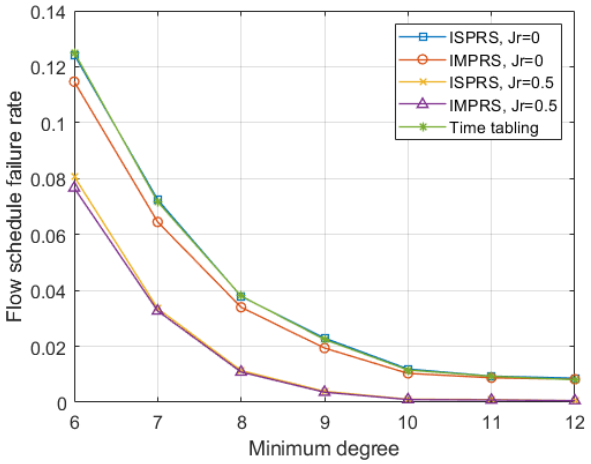


FIGURE 9. Flow schedule failure rates with respect to the minimum degree with mean frame size 1000 bytes.

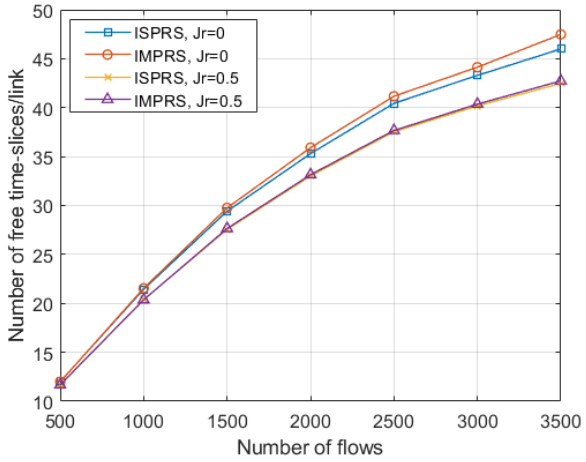


FIGURE 10. Average number of free time-slices of a link with respect to the number of flows.

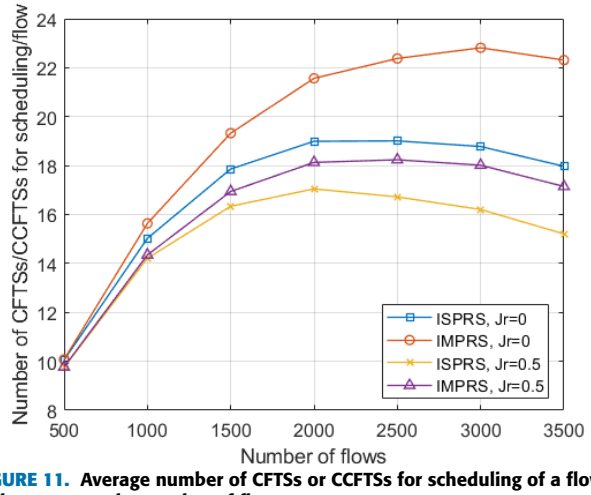


FIGURE 11. Average number of CFTSs or CCFTSs for scheduling of a flow with respect to the number of flows.

failure rates are high, but the difference diminishes as jitter bound size increases. The improvement due to nonzero jitter bound ratio is more apparent when the flow schedule failure rate is larger. For small nonzero jitter bound ratios, decrease of failure rates due to nonzero jitter bound is larger and as jitter bound ratio increases, the reduction saturates. Since time-tabling method cannot support nonzero jitter, it is not shown.

Fig. 8 shows flow schedule failure rates with respect to mean frame size μ for two jitter bound ratios when the number of flows is 1000. As mean frame size increases, schedule failure rates increase, too. Scheduling performance with nonzero jitter bound ratio of 0.5 is much better than performance with zero jitter. Schedule failure rates of IMPRS are lower than ISPRS when mean frame size is larger and zero jitter is used. This means that with higher schedule failure rates IMPRS scheme is better than ISPRS. But with nonzero jitter, performances of ISPRS-NZJ and IMPRS-NZJ are similar. Time-tabling method shows performance almost equal to ISPRS with zero jitter.

Fig. 9 shows that flow schedule failure rates decrease as the minimum degree of a node increases. As minimum degree increases, the number of links of a network increases. Thus there are a larger number of shortest paths for a flow and average hop count of shortest paths for a flow decreases. With smaller hop count of a path, it is easier to find free time-slices common in the links of the path. Thus flow scheduling is more likely to succeed with higher degree.

Fig. 10 shows that average number of free time-slices of a link increases almost linearly at first but increase rate diminishes as the number of flows increases. By allocation of a free time-slice to a subflow, the free time-slice could be divided into maximum two free time-slices. As the number of scheduled flows increases, average duration of a free time-slice of a link decreases and an allocated free time-slice is less likely to be divided into two free time-slices. Thus the increase rate of the number of free time-slices in a link diminishes as the number of flows increases.

Fig. 11 shows the average number of CFTSs in the last hop link used in ISPRS schemes and CCFTS at the destination node in IMPRS schemes, when the number of flows increases. With larger number of flows, duration of a free time-slice in a link is more likely to be small and it is less likely that an intersection of two free time-slices is long enough for frame duration of a flow. Thus when the number of flows is small, average numbers of CFTSs and CCFTSs increase with the number of flows. But as the number of flows increases more, they saturate and even tend to decrease. The number of CCFTSs is larger than the number of CFTSs, because CCFTSs are obtained by union of CFTSs of several links.

Fig. 12 shows schedule failure rates for various numbers of flows. As the number of flows increases, schedule failure rates increase. When the number of flows increases, the number of CFTSs or CCFTSs for scheduling saturates as shown in Fig. 11 and average duration of each CFTS or CCFTS decreases. Thus the probability to find feasible time-slices common in the links of a path decreases and schedule failure rate increases.

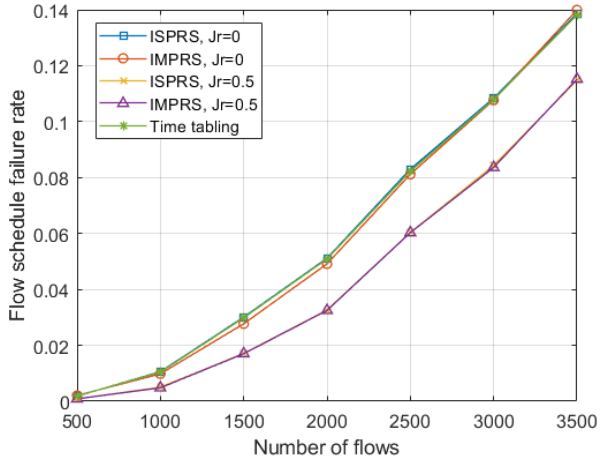


FIGURE 12. Flow schedule failure rates with respect to the number of flows.

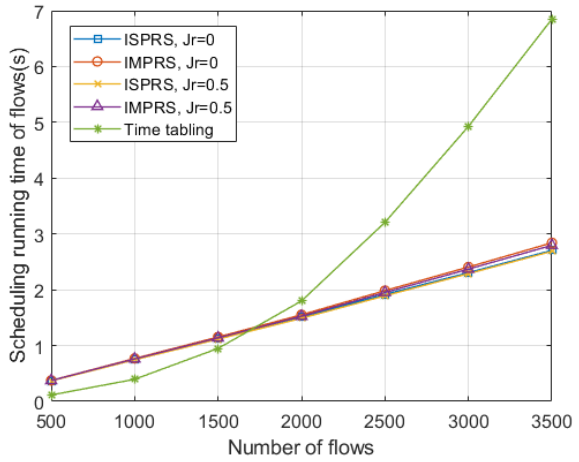


FIGURE 13. Schedule running time with respect to the number of flows.

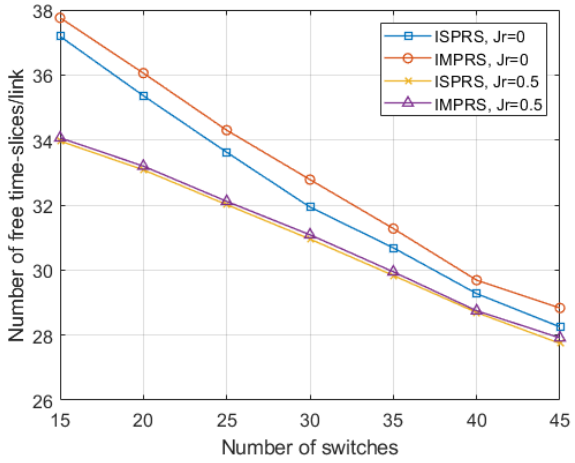


FIGURE 14. Average number of free time-slices of a link with respect to the number of switches when the number of flows is 2000.

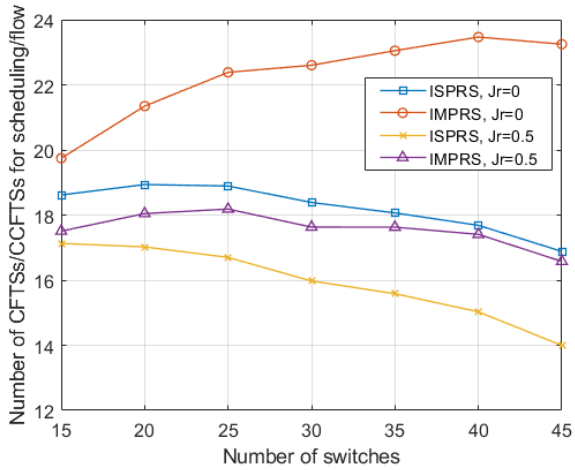


FIGURE 15. Average number of CFTSs or CCFTSs for scheduling of a flow with respect to the number of switches when the number of flows is 2000.

Fig. 13 shows average schedule running times for various numbers of flows. Schedule running time is the running time of performing routing and scheduling for all the flows given. As the number of flows increases, schedule running times increase. Both ISPRS and IMPRS schemes show similar schedule running times and jitter bound ratio size affects schedule running times little. Schedule running time of time-tabling method shows a large difference from the proposed schemes. As flow number increases, schedule running times of the proposed schemes increase almost linearly, while schedule running time of the time-tabling method increases much faster than the proposed schemes. Time complexity of the time-tabling method is stated as $O(N^3)$ with N flows in [11] and the simulation shows similar results.

Although time complexities of the proposed schemes are $O(N^2)$, schedule running times show almost linear complexity in Fig. 13. The proposed scheduling schemes require searching free time-slices in links. Since the number of free time-slices in a link is $O(N)$, searching for scheduling requires $O(N)$ reading operations for a flow and $O(N^2)$ reading operations for all the flows. As for required writing operations, update of free time-slice information of links with scheduling result of a flow requires $O(1)$ updates. However, to maintain the information sorted, $O(N)$ shifting and writing operations are required. Also derivation of CFTS or CCFTS sets also requires $O(1)$ writing operations since the number of CFTSs in a link or CCFTSs of a node is $O(1)$ as shown in Fig. 11, when the number of flows is large. Thus overall required writing operation is $O(N)$ for scheduling of a flow and $O(N^2)$ for scheduling of N flows. Meanwhile, analysis of running time of the simulator reveals that initial memory access for writing consumes much larger time than memory writing and reading operations. For example, for update of a free time-slice information, time for first accessing the array registering the information for writing is very high but shifting the elements of array and inserting a new information require less time. Thus in most cases, scheduling running time is dominated by initial memory access time for writing. Since initial memory access for writing occurs $O(1)$ for scheduling of a flow in the proposed schemes, scheduling running times of all the flows are shown to increase linearly as the number of flows increases in Fig. 13.

Fig. 14 shows the average number of free time-slices of a link after all the flows are scheduled when the number of switches changes and the number of flows is 2000. As the

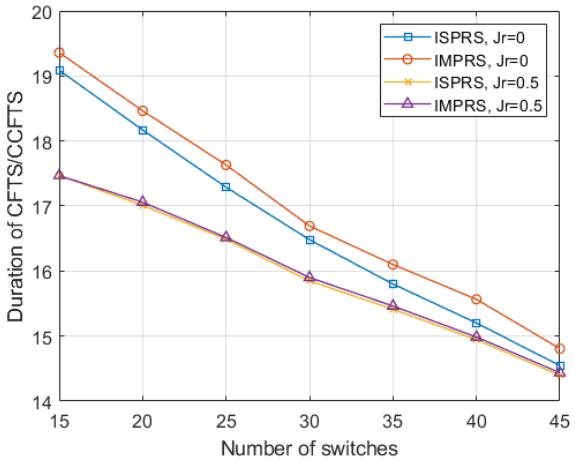


FIGURE 16. Average duration of a CFTS or CCFTS in basic time unit for scheduling of a flow with respect to the number of switches when the number of flows is 2000.

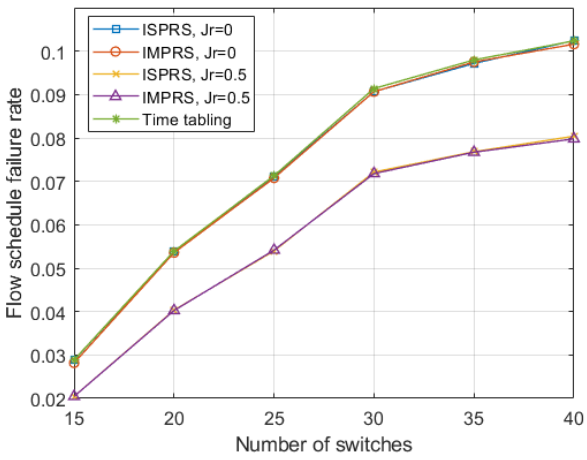


FIGURE 17. Flow schedule failure rates with respect to the number of switches when the number of flows is 2000.

number of switches increases, the number of links of a network and average hop count of paths increase. Large number of links results in decrease of flows served in a link, while large hop count of a path means that a flow causes more traffic in a network. These two conflicting effects result in decreased number of free time-slices in a link as the number of switches increases. It is due to the fact that increase of link is larger than increase of hop count of paths. Smaller number of free time-slices means that less number of flows are served in a link. Thus small number of free time-slices of a link means that average duration of free time-slices is large.

Fig. 15 shows the average numbers of CFTSs in the last hop link in ISPRS scheme and CCFTS at the destination node in IMPRS scheme, as the number of switches increases. It is shown that the numbers of CFTSs and CCFTSs do not vary much as the number of switches increases. Although the number of free time-slices in a link decreases as shown in Fig. 14 as the number of switches increases, the numbers of CFTSs and CCFTSs do not decrease much. It is because average duration of free time-slices is large and thus it is more likely that delayed intersection of two time-slices in consecutive links produce feasible CFTSs.

Fig. 16 shows the average duration of CFTSs in the last hop link in ISPRS scheme or CCFTSs in the destination node in IMPRS scheme with respect to the number of switches. As the number of switches increases, average durations of CFTSs and CCFTSs decrease. Note that as the number of switches increases, average hop count increases. CFTSs are actually intersection of the free time-slices in all the links of a path. As a result of intersection, the resulting free time-slices are likely to be shortened. Thus with larger hop count, durations of resulting CFTSs are reduced more. Since CCFTS is union of CFTSs, small duration of CFTS results in likewise small duration of CCFTS.

Fig. 17 shows schedule failure rates with respect to the number of switches when the number of flows is 2000. As the number of switches increases, flow schedule failure rates of all the considered schemes increase with diminishing increase rates. Although each link has less flows to serve with larger number of switches, schedule failure rates increases. This is due to the fact that with larger number of switches, average hop count of a path increases and average durations of CFTSs and CCFTSs decrease as in Fig. 16. Note that IMPRS scheme has similar performance to ISPRS scheme. For multipath routing to be effective, there should be a large number of shortest paths for a flow, which corresponds to a large hop count of the paths. However, as hop count of a path increases, the probability to find feasible free time-slices common in all the links decreases. Thus, increase of network size does not lead directly to performance improvement by multipath scheduling.

VII. CONCLUSION

In this work, ISPRS-NZJ and IMPRS-NZJ schemes are proposed in IEEE 802.1 Qbv TAS system. With the proposed schemes, the effects of multipath routing and nonzero jitter bound on the scheduling performance are studied. Both nonzero jitter scheduling and multipath scheduling features could provide more flexibility in scheduling and thus they could reduce schedule failure rates. Scheduling using nonzero jitter bound decreases schedule failure rates more apparently than multipath routing. The proposed schemes rely on frequent memory writing operations of information on free time-slices. Thus schedule running time could be affected by memory access efficiency.

TAS scheduling with multipath routing and nonzero jitter bound have been rarely studied. More works could be done to improve the proposed schemes and to explore the application of the proposed approach. For example, the proposed approach of deriving free time-slices common in the links could be extended to multicast in TSN, which is another important application area in TSN but has not been studied much. Also, nonzero jitter bound and multipath routing could be applied for TAS scheduling not using no-wait switching.

REFERENCES

[1] *IEEE 802.1Qbv—Enhancements for Scheduled Traffic*, IEEE Standard 802.1 TSN Task Group, [Online]. Available: http://www.ieee802.org/1/pages/802.1bv.html

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

[3] D. Theiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers," in *Proc. IEEE Veh. Netw. Conf.*, Dec. 2015, pp. 251–258.

[4] A. A. Syed, S. Ayaz, T. Leinmuller, and M. Chandra, "MIP-based joint scheduling and routing with load balancing for TSN based in-vehicle networks," in *Proc. IEEE Veh. Netw. Conf.*, Dec. 2020, pp. 1–7.

[5] T. Steinbach, H.-T. Lim, F. Korf, T. C. Schmidt, D. Herrscher, and A. Wolisz, "Tomorrow's in-car interconnect? A competitive evaluation of IEEE 802.1 AVB and time-triggered Ethernet (AS6802)," in *Proc. IEEE Veh. Technol. Conf. (VTC Fall)*, Sep. 2012, pp. 1–5.

[6] L. Silva, P. Pedreiras, P. Fonseca, and L. Almeida, "On the adequacy of SDN and TSN for industry 4.0," in *Proc. IEEE 22nd Int. Symp. Real-Time Distrib. Comput. (ISORC)*, May 2019, pp. 43–51.

[7] A. Nasrallah, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 88–145, 1st Quart., 2019.

[8] Y. Liu, Y. Zhou, J. Yuan, and L. Liu, "Delay aware flow scheduling for time sensitive fronthaul networks in centralized radio access network," *IEEE Trans. Commun.*, vol. 68, no. 5, pp. 2992–3009, May 2020.

[9] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. 24th Int. Conf. Real-Time Netw. Syst.*, Oct. 2016, pp. 183–192.

[10] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," in *Proc. 31st IEEE Real-Time Syst. Symp.*, Nov. 2010, pp. 375–384.

[11] 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.*, Oct. 2016, pp. 203–212.

[12] N. G. Nayak, F. Dürr, and K. Rothermel, "Incremental flow scheduling and routing in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 5, pp. 2066–2075, May 2018.

[13] Y. Huang, S. Wang, T. Huang, B. Wu, Y. Wu, and Y. Liu, "Online routing and scheduling for time-senstive networks," in *Proc. IEEE ICDCS*, Jul. 2021, pp. 272–281.

[14] D. Allan, P. Ashwood-Smith, N. Bragg, J. Farkas, D. Fedyk, M. Ouellete, M. Seaman, and P. Unbehagen, "Shortest path bridging: Efficient control of larger Ethernet networks," *IEEE Commun. Mag.*, vol. 48, no. 10, pp. 128–135, Oct. 2010.

[15] D. Allan, J. Farkas, and S. Mansfield, "Intelligent load balancing for shortest path bridging," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 163–167, Jul. 2012.

[16] I. Cidon, R. Rom, and Y. Shavitt, "Analysis of multi-path routing," *IEEE/ACM Trans. Netw.*, vol. 7, no. 6, pp. 885–896, Dec. 1999.

[17] J. Huang, J. O. Blech, A. Raabe, C. Buckl, and A. Knoll, "Static scheduling of a time-triggered network-on-chip based on SMT solving," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2012, pp. 509–514.

[18] Y. Nakayama, R. Yaegashi, A. H. N. Nguyen, and Y. Hara-Azumi, "Real-time reconfiguration of time-aware shaper for ULL transmission in dynamic conditions," *IEEE Access*, vol. 9, pp. 115246–115255, 2021.

[19] X. Jin, C. Xia, N. Guan, C. Xu, D. Li, Y. Yin, and P. Zeng, "Real-time scheduling of massive data in time sensitive networks with a limited number of schedule entries," *IEEE Access*, vol. 8, pp. 6751–6767, 2020.

[20] M. Vlk, Z. Hanzalek, K. Brejchova, S. Tang, S. Bhattacharjee, and S. Fu, "Enhancing schedulability and throughput of time-triggered traffic in IEEE 802.1Qbv time-sensitive networks," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 7023–7038, Nov. 2020.

[21] D. Bujosa, M. Ashjaei, A. V. Papadopoulos, T. Nolte, and J. Proenza, "HERMES: Heuristic multi-queue scheduler for TSN time-triggered traffic with zero reception jitter capabilities," in *Proc. 30th Int. Conf. Real-Time Netw. Syst.*, Jun. 2022, pp. 70–80.

[22] V. Gavriluţ, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, pp. 75229–75243, 2018, doi: 10.1109/ACCESS.2018.2883644.

[23] A. Berisa, L. Zhao, S. S. Craciunas, M. Ashjaei, S. Mubeen, M. Daneshtalab, and M. Sjödin, "AVB-aware routing and scheduling for critical traffic in time-sensitive networks with preemption," in *Proc. 30th Int. Conf. Real-Time Netw. Syst.*, Jun. 2022, pp. 207–218.

[24] M. Vlk, K. Brejchová, Z. Hanzálek, and S. Tang, "Large-scale periodic scheduling in time-sensitive networks," *Comput. Oper. Res.*, vol. 137, Jan. 2022, Art. no. 105512.

[25] Z. Pang, X. Huang, Z. Li, S. Zhang, Y. Xu, H. Wan, and X. Zhao, "Flow scheduling for conflict-free network updates in time-sensitive software-defined networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 3, pp. 1668–1678, Mar. 2021.

[26] N. Wang, Q. Yu, H. Wan, X. Song, and X. Zhao, "Adaptive scheduling for multicluster time-triggered train communication networks," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 1120–1130, Feb. 2019.

[27] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Routing and scheduling of time-triggered traffic in time-sensitive networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4525–4534, Jul. 2020.

[28] A. Arestova, W. Baron, K.-S.-J. Hielscher, and R. German, "ITANS: Incremental task and network scheduling for time-sensitive networks," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 369–387, 2022.

[29] L. Xu, Q. Xu, J. Tu, J. Zhang, Y. Zhang, C. Chen, and X. Guan, "Learning-based scalable scheduling and routing co-design with stream similarity partitioning for time-sensitive networking," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13353–13363, Aug. 2022.

[30] A. Alnajim, S. Salehi, and C.-C. Shen, "Incremental path-selection and scheduling for time-sensitive networks," in *Proc. IEEE GLOBECOM*, Dec. 2019, pp. 1–6.

[31] M. Vlk, Z. Hanzálek, and S. Tang, "Constraint programming approaches to joint routing and scheduling in time-sensitive networks," *Comput. Ind. Eng.*, vol. 157, Jul. 2021, Art. no. 107317.

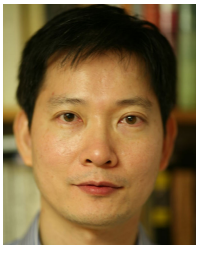
[32] L. Leonardi, L. L. Bello, and G. Patti, "Exploiting software-defined networking to improve runtime reconfigurability of TSN-based networks," in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2022, pp. 1–4.

[33] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements*, IEEE Standard 802.1Qcc-2018 (Amendment to IEEE Standard 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018), pp. 1–208, Oct. 2018.

[34] H. Chahed and A. J. Kassler, "Software-defined time sensitive networks configuration and management," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2021, pp. 124–128.

[35] I. Alvarez, A. Servera, J. Proenza, M. Ashjaei, and S. Mubeen, "Implementing a first CNC for scheduling and configuring TSN networks," in *Proc. IEEE 27th Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2022, pp. 1–4.

[36] *Time and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, IEEE Standard 802.1AS-2020, 2020.



DONG-JUN LEE (Member, IEEE) received the Ph.D. degree in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2000. From 2000 to 2005, he was with Samsung Electronics Company Ltd., South Korea. Since 2005, he has been with Korea Aerospace University, Goyang, South Korea, where he is currently a Professor. His research interests include resource management in wireless networks, location-based service, sensor networks, and time-sensitive networks.