Received February 16, 2019, accepted April 22, 2019, date of publication May 14, 2019, date of current version June 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2916651

Flit Scheduling for Cut-Through Switching: Towards Near-Zero End-to-End Latency

KYUBO SHIN, SEOKWOO CHOI, AND HYOIL KIM, (Senior Member, IEEE)
School of ECE, Ulsan National Institute of Science and Technology, Ulsan 44919, South Korea

Corresponding author: Hyoil Kim (hkim@unist.ac.kr)

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean Government (MSIP) (Research on Near-Zero Latency Network for 5G Immersive Service) under Grant 2015-0-00278, and in part by the Research Funding from Ulsan National Institute of Science and Technology (UNIST) under Grant 1.180065.01.

ABSTRACT Achieving low end-to-end latency with high reliability is one of the key objectives for future mission-critical applications, like the Tactile Internet and real-time interactive Virtual/Augmented Reality (VR/AR). To serve the purpose, cut-through (CT) switching is a promising approach to significantly reduce the transmission delay of store-and-forward switching, via flit-ization of a packet and concurrent forwarding of the flits belonging to the same packet. CT switching, however, has been applied only to well-controlled scenarios like network-on-chip and data center networks, and hence flit scheduling in heterogeneous environments (e.g., the Internet and wide area network) has been given little attention. This paper tries to fill the gap to facilitate the adoption of CT switching in the general-purpose data networks. In particular, we first introduce a packet discarding technique that sheds the packet expected to violate its delay requirement and then propose two flit scheduling algorithms, *f* EDF (flit-based Earliest Deadline First) and *f* SPF (flit-based Shortest Processing-time First), aiming at enhancing both reliability and end-to-end latency. Considering packet delivery ratio (PDR) as a reliability metric, we performed extensive simulations to show that the proposed scheduling algorithms can enhance PDR by up to 30.11% (when the delay requirement is 7 ms) and the average end-to-end latency by up to 13.86% (when the delay requirement is 10 ms), against first-in first-out (FIFO) scheduling.

INDEX TERMS Computer networks, cut-through switching, end-to-end latency, packet switching, performance evaluation, scheduling algorithm.

I. INTRODUCTION

Achieving low end-to-end latency with high reliability is key to the success of future communication networks and services. For example, 5G (Fifth Generation) communications has set medium access latency of 1 millisecond as one of its main design goals [1]. The Tactile Internet [2] requires end-to-end latency of few milliseconds over the Internet to deliver time-critical sensory data for online tactile applications like remote haptic control. In addition, real-time immersive services based on Virtual/Augmented Reality (VR/AR) would need to let numerous users interact with each other and with surrounding IoT (Internet of Things) devices with very low latency.

To realize near-zero end-to-end latency, mitigating per-router transmission delay is most critical. In modern data communication networks like the Internet, end-to-end latency is broken down into transmission delay, queueing delay, propagation delay, and processing delay, among which transmission and queueing delays are most significant. In particular, heavy transmission delay is the consequence of today's store-and-forward (SF) switching based network architecture, which is not scalable with the number of forwarding hops since per-hop packet transmission delay is accumulated at every router.

Cut-through (CT) switching is a promising technique to significantly reduce the transmission delay of SF switching. CT switching divides a packet into multiple flits, and concurrently forwards them from the ingress link to the egress link [3]. Fig. 1(b) illustrates the concept of CT switching, where a flit of a packet is forwarded to the downstream switch (or router) while its following flit is being received from the upstream switch simultaneously. Accordingly, CT switching achieves significantly reduced end-of-end latency compared to SF switching, as shown in Fig. 1.

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

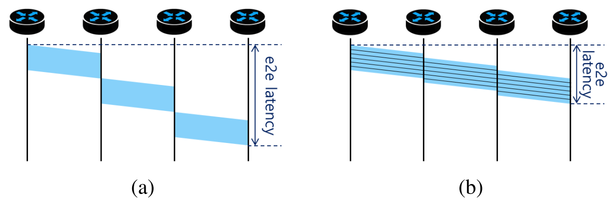


FIGURE 1. End-to-end transmission delay: SF switching vs. CT switching. (a) SF switching. (b) CT switching.

Flit scheduling is crucial to successfully adopting CT switching in general-purpose data networks. Traditionally, flit scheduling was not actively studied because CT switching has been mostly adopted in well-structured networks like network-on-chip (NoC) [4] and data center networks (DCN) [5] where simple and deterministic scheduling works well, e.g., transmitting the flits of a packet sequentially and contiguously. On the contrary, most general-purpose data networks consist of heterogeneous links and random topology, and thus multiple sessions are multiplexed together in a random manner. As a result, flits can experience queueing delay at congested routers which should be properly managed by flit scheduling so as not to weaken the benefit of CT switching.

Unlike classical packet scheduling, flit scheduling for CT switching confronts a unique challenge: a flit of a packet can be forwarded to the next router before the whole packet arrives at a router. Moreover, the arrival times of the flits are hard to predict which incurs additional complexity in estimating per-hop and end-to-end latency. Nevertheless, flit scheduling has not been given enough attention in the context of noncontiguously arriving flits and multiplexed sessions, and this paper tries to fill the gap.

The contribution of this paper is three-fold. First, we propose a packet discarding mechanism to enhance network utilization, which drops the packet expected to violate its delay requirement. Next, we propose two flit scheduling algorithms, *f*EDF (flit-based Earliest Deadline First) and *f*SPF (flit-based Shortest Processing-time First), which are inspired by two classical CPU schedulers, Earliest Deadline First (EDF) and Shortest Job First (SJF), respectively. Our proposal has novelty in the sense that we have completely re-engineered the classical schemes to fit into the framework of flit-based CT switching, which is totally different from CPU scheduling. Moreover, each proposed algorithm is further classified into two types, *to-destination* and *per-hop*, depending on whether the scheduler considers the overall delay requirement or the per-hop delay budget. Via the proposed schemes, we aim to achieve low end-to-end latency with high reliability, where the reliability is measured by the ratio of packets satisfying the delay requirement. Finally, we evaluate the performance of the proposed flit schedulers, to reveal how much performance they can achieve in terms of the average end-to-end latency and reliability, and which of them is superior.

The rest of the paper is organized as follows. Section II overviews related work, and Section III defines our system model. Section IV introduces our packet discarding mechanism and two proposed flit scheduling algorithms. Then, Section V evaluates the performance enhancement achieved by the proposed algorithms via extensive simulations. Finally, the paper concludes with Section VI.

II. RELATED WORK

There have been several QoS (Quality-of-Service) protocols designed to serve low-latency services in data communication networks [6]. In IntServ (Integrated Services), each time-sensitive traffic flow reserves end-to-end network resources via the Resource reSerVation Protocol (RSVP). In DiffServ (Differentiated Services), packets are classified into different QoS classes, and each packet's per hop behaviors (e.g., packet scheduling and policing) are determined accordingly. In addition, Software Defined Networking (SDN) tries to support end-to-end QoS provisioning by separating the control plane from the forwarding plane and making the former to perform traffic control in a centralized manner [7].

Our proposal, however, deals with low-latency traffic differently from IntServ, DiffServ, and SDN. While IntServ reserves network resources for each session, our flit scheduling does not maintain per-session resources but schedules the queued flits from multiplexed sessions dynamically. Unlike DiffServ where the packets in the same QoS class are served equally in a router, our schedulers are aware of per-packet delay requirement and accumulated delay along the routing path, and act adaptively. Finally, while SDN needs a centralized controller to schedule packets, our schemes can operate in a distributed manner at each router.

Traditionally, CT switching has been used mostly in DCN [5] and NoC [4]. In DCN, various flow scheduling algorithms were proposed to enhance end-to-end latency and the packet loss ratio in well-defined topologies [8], while our work deals with CT switching in a random and heterogeneous topology. For example, PDQ (Preemptive Distributed Quick) proposed in [9] schedules each flow using the scheduling header embedded in packets. In [10], pFabric is proposed to reduce the average flow completion time by applying priority-based flow scheduling at switches and rate control at end hosts. The aforementioned studies, however, are built upon the topologies developed specifically for DCN such as Fat-tree [11] and BCube [12].

Unlike DCN that focused on flow scheduling, studies in NoC have focused on flit scheduling. Reference [13] analyzed the per-flow end-to-end delay bounds of SPQ (Strict Priority Queueing) and WRR (Weighted Round Robin) scheduling, using network calculus. ValadBeigi *et al.* [14] proposed a flit scheduling algorithm to support their proposed NoC architecture, which aims to reduce latency, energy consumption, and area overhead. In addition, [15] proposed non-preemptive regions for flit-level preemptive scheduling to support real-time traffic, where a non-preemptive region specifies beyond which flit of a packet its remaining flits should be forwarded exclusively without being interleaved with other packets. The paper, however, assumes that every flow has its own priority class (instead of being mapped to one of finite priority classes), which is unrealistic. The buffer-based adaptive round-robin scheduling algorithm presented in [16] utilizes the buffer state of downstream routers to reduce the chance of congestion. Such studies in NoC, however, assumed not only a small buffer size at each router but also a well-defined topology, mainly 2D-mesh, which are only valid in the NoC environment.

There exists only few works on CT switching in general-purpose data networks, among which Myrinet is a gigabit LAN (Local Area Network) that aims at reducing the end-to-end delay [17] and Autonet is designed to reduce its switching latency. [18]. Additionally, Time-Triggered Ethernet, a variant of the classical Ethernet, switches all Time-Triggered messages according to the CT mechanism [19]. Aforementioned approaches, however, did not consider flit scheduling as well since the classical CT switching model is restricted to the cases with no session multiplexing, as found in virtual CT switching (VCTS) [20], quasi CT switching (QCTS) [21], and general CT switching (GCTS) [22].

On the other hand, asynchronous transfer mode (ATM) encodes data traffic into a stream of fixed-sized packets, also known as *cells* [23]. Even though ATM resembles CT switching in the sense that it transmits fixed length data units (e.g., cells), an ATM cell in reality is quite different from a flit. While each cell in ATM is routed independently, a flit in CT switching is a part of a packet and thus is forwarded (possibly non-contiguously) as a group with other flits from the same packet. Although some studies on ATM dealt with a group of cells, there still exists wide discrepancy between them and our problem. For instance, [24] applied FEC (forward error correction) coding to a group of cells and considered their end-to-end delay, but the cells were not related to each other. [25] proposed iSLIP, an iterative and round-robin scheduling algorithm for ATM switches aiming at avoiding head-of-line blocking and starvation at input queues. Although originally designed for ATM, iSLIP has been also widely used for DCN (e.g., Cisco Nexus 5000 series [26]) and for NoC [16], [27]. In Section V, we will use iSLIP as one of reference algorithms to compare with our proposed algorithms, not only due to its popularity (in ATM, DCN, and NOC), but also due to its generality to fit into a random topology like ATM.

Our proposed CT-based flit scheduling mechanism aims at dealing with both session multiplexing and flit scheduling, motivated by the following aspects. First, we consider a general network model with heterogeneous link rates and a random network topology, where inter-session multiplexing is inevitable. Second, guaranteeing low per-packet end-to-end latency requires consideration to when and how a group of flits belonging to the same packet are forwarded at each router. Lastly, depending on various session multiplexing and flit arriving scenarios, it is necessary to interleave flits from different packets.

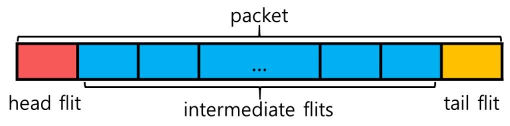


FIGURE 2. Flitization of a packet.

III. SYSTEM MODEL

This paper focuses on near-zero-latency (NZL) sessions that require satisfying a hard end-to-end delay constraint per packet, which is much smaller than what today's Internet protocols offer. To achieve the goal, we adopt CT switching at each router[1] of the NZL-serving network, where the routers may provide both CT and SF switching (unlike NoC and DCN that are dedicated to CT switching). In CT switching, a packet is divided into a head flit, multiple intermediate flits, and a tail flit, as shown in Fig. 2. Accordingly, packet forwarding at a router is performed flit-by-flit as in Fig. 1, which significantly reduces the transmission delay of the traditional SF switching thus leading to much smaller end-to-end latency. To be more specific, denoting single-hop packet transmission time by P_{sp}, single-hop flit transmission time by F_{sp}, and the number of hops by N_{hops}, the packet's end-to-end transmission delay can be reduced *from* $P_{sp} \cdot N_{hops}$ *to* $P_{sp} + F_{sp} \cdot (N_{hops} - 1)$, if no queueing occurs.

For further promoting delay guarantee of NZL traffic, we assume that a portion of network resources is dynamically allocated to NZL sessions and new arriving NZL sessions undergo admission control at their source routers.[2] The portion of NZL resources can be adjusted considering the balance between NZL traffic and classical (e.g., best-effort) traffic. Therefore, in this paper, a router's input and output link rates are the consequence of such resource allocation, and thus they might be much smaller than their physical capacity.

A. NZL PACKETS AND FLIT FORWARDING

Let us consider an NZL session S^i (i: session index) which is characterized by its packet arrival rate λ^i (packets/slot) and 'hard' delay requirement D^i_{req} (slots).[3] Then, an NZL packet belonging to S^i, denoted by $P^{i,k}$ (k: packet index), is first flit-ized at the source router, and then the flits of the packet are transmitted and forwarded in order through the packet's pre-determined routing path. It is assumed that every routing path has the path MTU (Maximum Transmission Unit) of 1,500 (bytes) at the Internet Protocol (IP) layer.[4] Therefore, the packet size of $P^{i,k}$ in the IP layer, denoted by $s^{i,k}_p$ (bytes), is upperbounded by 1,500 (bytes).

[1]This paper uses 'switch' and 'router' interchangeably, while focusing on switching and forwarding of NZL packets/flits.

[2]The design of resource allocation and admission control is out of the scope of this paper, and left as our future work.

[3]Session information can be informed to the routers on the routing path using existing message protocols like ICMP (Internet Control Message Protocol). Note that the related protocol design is out of the scope of this paper.

[4]1,500 bytes is the Ethernet-based MTU.

At a router with packet $P^{i,k}$ under processing,[5] $l^{i,k}_{path}$ denotes the number of remaining links from the router to the destination, where each link is denoted by $L^{i,k}_x$, $x \in \{1, 2, \cdots, l^{i,k}_{path}\}$ in the order of its proximity to the current router. The propagation delay and the link rate of $L^{i,k}_x$ are denoted by $d^{i,k}_x$ and $\rho^{i,k}_x$, respectively. We denote by $r^{i,k}_0$ the packet reception rate from the packet's input link. In addition, $d^{i,k}_x$, $\rho^{i,k}_x$, $r^{i,k}_0$ are assumed known to the router.[6]

The *end-to-end latency* of a packet is defined as the difference between the departure time of its head flit at the source router and the arrival time of its tail flit at the destination router. For any packet $P^{i,k}$, their delay requirement is equal to D^i_{req} since we assume D^i_{req} is a session's characteristic, not a packet's. Thus, packet $P^{i,k}$'s deadline to arrive at the destination is determined as $T^{i,k}_d + D^i_{req}$ where $T^{i,k}_d$ (slots) denotes the departure time of packet $P^{i,k}$ at the source router. The *reliability* of the network is measured by the portion of packet transmissions successfully arriving at the destination within D^i_{req}, which is henceforth referred to as *packet delivery ratio* (PDR).

This paper considers PDR as a metric of network reliability, since CT switching in general-purpose heterogeneous networks confronts vastly different challenges than classical CT switching. For example, CT switching in NoC deals with a set of *schedulable* flows that can already satisfy the performance goals like end-to-end latency [28]. In our case, however, schedulability cannot be guaranteed due to the complex network topology and high randomness in packet arrivals and routing paths. In such a network, the ratio of packets delivered within deadline (i.e., PDR in our work) has been used as a performance metric of measuring how reliably a given network satisfies the latency requirement [29], [30]. Accordingly in the sequel, while the flits of a packet are forwarded in the network, if a router finds that the accumulated latency of the packet exceeds the given packet deadline, all the flits of such a packet are discarded.

B. FLITIZATION AND ROUTER ARCHITECTURE

We assume that a packet $P^{i,k}$ is flitized into multiple same-size flits as shown in Fig. 3, where the flit size (in the IP layer) is denoted by s_f (bytes). A flit consists of a flit header, followed by an IP header in case of the head flit, payload, and possibly zero padding in case of the tail flit. The flit header with size $s_h = 4$ (bytes) includes the flit preamble (0.5 bytes), the packet identifier (2 bytes), the flit's sequence number (0.5 bytes), and the expected remaining time $T^{i,k}_t$ until the arrival of the tail flit at the next downstream router (1 byte), where the estimation of $T^{i,k}_t$ will be discussed later in Section IV.[7] The flit size is lowerbounded as $s_f \geq 60 + s_h = 64$ (bytes), since the minimum flit size is determined by the minimal head flit consisting of only a flit header and an IP header, where the maximum IP header size is 60 (bytes).[8] In addition, the IP header in the head flit of $P^{i,k}$ is assumed to contain $T^{i,k}_d$ in the *options* field.[9] Then, the number of flits composing $P^{i,k}$ is given as $\lceil s^{i,k}_p/(s_f - s_h)\rceil$ (flits).[10]

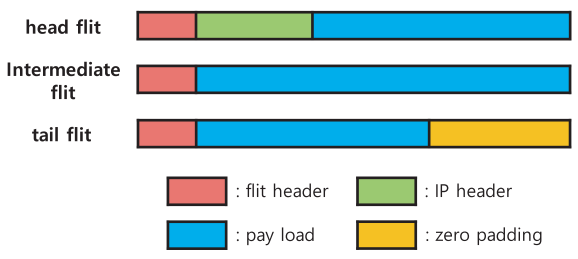


FIGURE 3. The flit structure.

Flitization incurs an extra overhead in packet transmission, since each flit requires a flit header and/or zero padded bits, and introduces additional encapsulation overhead in the medium access control (MAC) and physical (PHY) layers due to MAC/PHY headers, MAC trailer, etc. Therefore, as the flit size decreases, there is a tradeoff between a smaller per-hop transmission delay and an increased amount of the aforementioned flitization overhead. To quantify the impact of flitization, we define *flitization overhead* as the average number of bytes to transmit by CT switching *divided by* that of SF switching, both measured at the PHY layer for an IP packet, such as

$$\text{flitization overhead} = \frac{\sum_{\theta\in\Theta} \lceil \frac{\theta}{s_f - s_h} \rceil \cdot (s_f + s_o) \cdot p(\theta)}{\sum_{\theta\in\Theta} (\theta + s_o) \cdot p(\theta)}, \tag{1}$$

where θ (bytes) is an IP packet size, Θ is a set of all packet sizes found in the network, $p(\theta)$ is the ratio of packets in the network having the size of θ, and s_o (bytes) is the amount of extra information to append by the MAC and PHY layer frame encapsulation.

Fig. 4 shows the flitization overhead with varying flit size s_f, when $\Theta = \{1500\}$ and $s_o = 42$ (bytes).[11] In the figure, the local minima marked by red circles occur when $s_f = \lceil s^{i,k}_p/n\rceil + s_h, n = 2, 3, \cdots$, as proved by Theorem 1. It can be also seen that the flitization overhead grows exponentially fast as the flit size s_f decreases. Therefore, it is important to strike a balance between the flitization overhead and the transmission delay enhancement by CT switching, both are in a tradeoff relationship and rely on s_f.

[5]We intentionally omit the router index for notational simplicity.

[6]$d^{i,k}_x$, $\rho^{i,k}_x$, $r^{i,k}_0$ can also be informed to routers using existing message protocols like ICMP.

[7]The routers of the NZL-serving network are assumed to be able to differentiate packets from flits using the first 0.5 bytes of received packets/flits, which is the version field for an IP packet and the flit preamble for a flit.

[8]60 bytes is the maximum size specified by the *Internet Header Length (IHL)* field of the IPv4 header.

[9]We assume the options field contains an IP timestamp like in [31] but with more fine granularity, which may be achieved by developing future Internet RFCs (Request for Comments).

[10]In case $s^{i,k}_p$ is not a multiple of $(s_f - s_h)$, zero padding can be applied to the tail flit.

[11]According to IEEE 802.3 (i.e., Ethernet), s_o consists of (i) at the PHY layer: a 7 byte preamble, a 1 byte frame delimiter, and a 12 byte interpacket gap, and (ii) at the MAC layer: a 18 byte MAC header and a 4 byte frame check sequence.

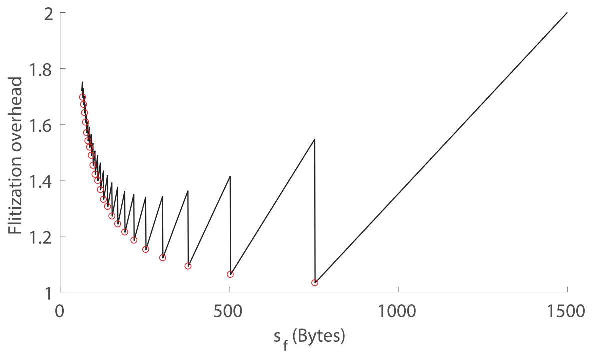


FIGURE 4. **Flitization overhead with varying flit size s_f, when the packet length is 1,500 bytes.**

To address the issue, Section V-A will determine the optimal s_f among the ones achieving the local minima of the flitization overhead.

Theorem 1: Assume $s_f \geq 64$ (bytes) and $s_h = 4$ (bytes). When all the packets in the network have the same size θ (bytes), $\theta \leq 1500$, the local minima of the flitization overhead in Eq. (1) *occur at $s_f = \lceil s_p^{i,k}/n \rceil + s_h, n = 2, 3, \cdots$.*

Proof: Please refer to the Appendix. ∎

Using s_f and s_o, we consider a slotted time model with the slot size $\tau = 8 \cdot (s_f + s_o)/L_\rho$ (sec), where L_ρ (bps) is the least common multiple of all link rates (in bps) in the network. Then, the time for transmitting a flit through a link becomes a multiple of τ. We assume synchronized time over network.[12]

In the sequel, the unit of time, rate, and packet size will be (slots), (flits/slot), and (flits), respectively, unless specified otherwise. Then, both the link rate and the transmission rate will be a value between 0 and 1 (in flits/slot), and the reciprocal of them becomes the number of slots to transmit a flit.

Each router in the NZL-serving network is assumed to be built upon a crossbar switch, where the number of input links, the number of output links, and the speedup of the crossbar switch are all equal. According to the assumption, a flit arriving at a router is instantly buffered at the output queue (OQ) of the destined output link, for which we assume a large enough buffer size not to experience an overflow. Then, the flits of a packet are classified into three categories: *forwarded* flits (to the next downstream router), *queued* flits at the OQ (the number of which is denoted by $n_q^{i,k}$), and *to-be-arriving* flits (the number of which is denoted by $n_r^{i,k}$). Fig. 5 illustrates the router architecture considered in this paper.

Frequently-used notations are summarized in Table 1.

IV. PROPOSED FLIT SCHEDULING ALGORITHMS

In this section, we first introduce the mechanism for discarding packets that are expected to violate the deadline, in order to enhance network utilization. Then, we propose two flit-scheduling algorithms, fEDF and fSPF, which aim to achieve low per-packet end-to-end latency and high reliability. Note that we have designed the algorithms to use only the current OQ's state and minimal topology information (e.g., link rates and propagation delays), because the network dynamics is very random and thus the status of the downstream routers is hard to predict.

Fig. 6 illustrates the flowchart of the proposed methodology. Whenever an OQ becomes able to transmit a flit (i.e., the OQ's output link becomes idle while the OQ is backlogged), the OQ conducts the packet discarding mechanism for every packet in the queue, and then chooses the flit to transmit (among the queued ones) according to fEDF or fSPF. Note that the time complexity of the three schemes will be discussed in each of the following sections.

[12]Time synchronization among switches can be achieved by various standardized methods like the Precision Time Protocol (PTP) [32]. The implementation issue of time synchronization is out of the scope of this paper.

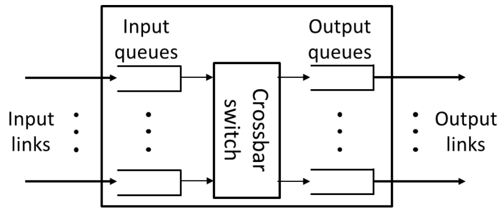


FIGURE 5. **Router architecture considered in the paper.**

TABLE 1. **A summary of notations.**

Notation	Definition
S^i	session i
λ^i (packets/slot)	packet arrival rate of session S^i
D^i_{req} (slots)	delay requirement of S^i
$P^{i,k}$	k-th packet of S^i
$T_d^{i,k}$ (slots)	departure time of $P^{i,k}$ at the source router
$l_{path}^{i,k}$	the number of the remaining links of $P^{i,k}$ from the current router to the destination
$L_x^{i,k}$	x-th remaining link of $P^{i,k}$
$\rho_x^{i,k}$ (flits/slot)	link rate of $L_x^{i,k}$
$d_x^{i,k}$ (slots)	propagation delay of $L_x^{i,k}$
$r_0^{i,k}$ (flits/slot)	reception rate of $P^{i,k}$ from the input link
$T_t^{i,k}$ (slots)	expected remaining time until the arrival of $P^{i,k}$'s tail flit at current router
$n_q^{i,k}$ (flits)	number of queued flits of $P^{i,k}$ at current OQ
$n_r^{i,k}$ (flits)	number of to-be-arriving flits of $P^{i,k}$ at current OQ
L_ρ (bps)	least common multiple of all link rates in the network

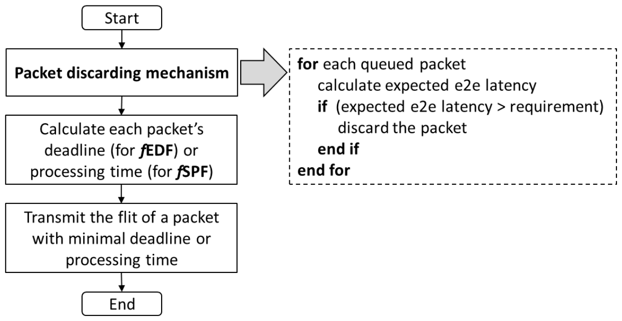

Start
Packet discarding mechanism
Calculate each packet's deadline (for fEDF) or processing time (for fSPF)
Transmit the flit of a packet with minimal deadline or processing time
End
for each queued packet
calculate expected e2e latency
if (expected e2e latency > requirement)
discard the packet
end if
end for

FIGURE 6. **Flowchart of our proposed scheduling algorithms.**

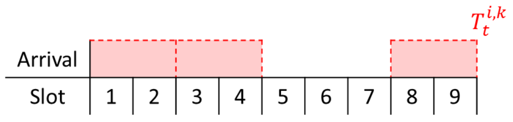


FIGURE 7. Expected arrival of to-be-arriving flits based on Condition 2, when $T_t^{i,k} = 9$, $n_r^{i,k} = 3$, and $r_0^{i,k} = 1/2$.

A. PACKET DISCARDING MECHANISM

We propose discarding the packet that cannot reach the destination by its deadline, so as to avoid wasting network resources to serve it unnecessarily. Specifically, packet $P^{i,k}$ is discarded when it is classified as a *violated packet*, which is defined as the one with the expected end-to-end latency $D_{e2e}^{i,k}$ exceeding D_{req}^{i}. Note that the classification of violated packets is performed by each router in the routing path.

$D_{e2e}^{i,k}$ is the sum of $D_a^{i,k}$ and $D_r^{i,k}$, where $D_a^{i,k}$ denotes the accumulated latency of the packet (from the source router to the current router) and $D_r^{i,k}$ denotes the expected remaining latency until the packet finally arrives at the destination. $D_a^{i,k}$ is trivially determined by subtracting $T_d^{i,k}$ from the present time. $D_r^{i,k}$, however, is hard to estimate because knowing $T_t^{i,k}$ is not enough to expect the time when each to-be-arriving flit will arrive at a router. Therefore, we try to consider the minimum possible value of $D_r^{i,k}$ which is denoted by $\underline{D}_r^{i,k}$ with the following motivation: if a packet is classified as a violated packet even with the most optimistic (i.e., minimal) estimate of $D_{e2e}^{i,k}$, the packet should be discarded right away.

$\underline{D}_r^{i,k}$ can be derived by considering the following two conditions.

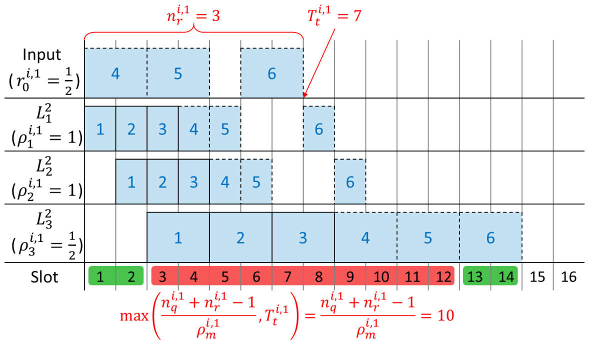
- **Condition 1**: The flits of a packet are not multiplexed by other flits at the current router and the downstream routers.
- **Condition 2**: For packet $P^{i,k}$, the to-be-arriving flits except the tail flit start to arrive at the current router sequentially with the reception rate $r_0^{i,k}$, and the tail flit arrives after $T_t^{i,k}$ slots from the current time.

Condition 1 implies that the packet will be transmitted exclusively in the remaining forwarding path, and Condition 2 considers the best possible scenario for the arrival pattern of the to-be-arriving flits except the tail flit. Hence, the two conditions lead to the minimal possible latency. Fig. 7 illustrates how the arrival time of to-be-arriving flits is predicted based on Condition 2: when $T_t^{i,k} = 9$ and $r_0^{i,k} = 1/2$, three to-be-arriving flits are expected to arrive after 2 slots, 4 slots, and 9 slots, respectively. Note that the tail flit should be predicted to arrive after $T_t^{i,k}$ slots, as informed $T_t^{i,k}$ by the upstream router.

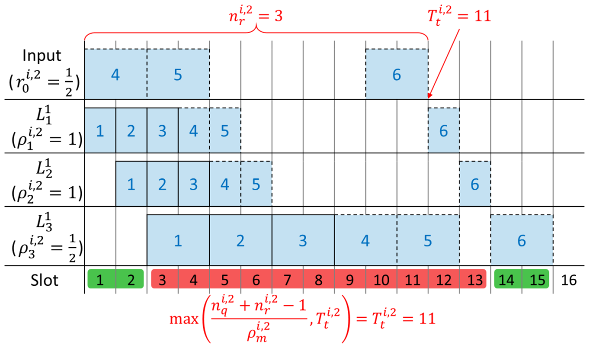
According to Conditions 1 and 2, $\underline{D}_r^{i,k}$ is calculated as

$$\underline{D}_r^{i,k} = \sum_{x=1}^{l_{path}^{i,k}} \left(\frac{1}{\rho_x^{i,k}} + d_x^{i,k} \right) + \max\left(\frac{n_q^{i,k} + n_r^{i,k} - 1}{\rho_m^{i,k}}, T_t^{i,k} \right), \tag{2}$$

where $\rho_m^{i,k}$ denotes the link rate of the lowest-rate link in the remaining path of $P^{i,k}$. Eq. (2) can be derived by adopting general link rates and propagation delay to the equation introduced in [15]. More specifically, the first term in Eq. (2) is the time it takes for the first queued flit in the current router to arrive at the last OQ in the routing path[13] (when $1 \le x \le l_{path}^{i,k} - 1$) *plus* the transmission and propagation time of the tail flit through the last link in the routing path (when $x = l_{path}^{i,k}$). In addition, the second term is the time it takes to transmit $(n_q^{i,k} + n_r^{i,k} - 1)$ flits (i.e., the flits excluding the tail flit) and to wait for the arrival of the tail flit at the OQ with the lowest-rate output link in the remaining routing path.


$n_r^{i,1} = 3$
$T_t^{i,1} = 7$
Input
$(r_0^{i,1} = \frac{1}{2})$
L_1^2
$(\rho_1^{i,1} = 1)$
L_2^2
$(\rho_2^{i,1} = 1)$
L_3^2
$(\rho_3^{i,1} = \frac{1}{2})$
Slot
$\max\left(\frac{n_q^{i,1} + n_r^{i,1} - 1}{\rho_m^{i,1}}, T_t^{i,1}\right) = \frac{n_q^{i,1} + n_r^{i,1} - 1}{\rho_m^{i,1}} = 10$

(a)


$n_r^{i,2} = 3$
$T_t^{i,2} = 11$
Input
$(r_0^{i,2} = \frac{1}{2})$
L_1^1
$(\rho_1^{i,2} = 1)$
L_2^1
$(\rho_2^{i,2} = 1)$
L_3^1
$(\rho_3^{i,2} = \frac{1}{2})$
Slot
$\max\left(\frac{n_q^{i,2} + n_r^{i,2} - 1}{\rho_m^{i,2}}, T_t^{i,2}\right) = T_t^{i,2} = 11$

(b)

FIGURE 8. $\underline{D}_r^{i,k}$ in two cases (solid box: Queued flits, dotted box: To-be-arriving flits). (a) $P^{i,1}$ with $T_t^{i,1} = 7$: $\underline{D}_r^{i,1} = 4 + 10 = 14$. (b) $P^{i,2}$ with $T_t^{i,2} = 11$: $\underline{D}_r^{i,2} = 4 + 11 = 15$.

For better understanding of Eq. (2), we provide an example in Fig. 8 where two packets $P^{i,1}$ and $P^{i,2}$ share the same routing path consisting of three remaining links (to their destination) with link rates 1, 1, 1/2, respectively. $P^{i,1}$ and $P^{i,2}$ also have $n_q^{i,1} = n_q^{i,2} = 3$, $n_r^{i,1} = n_r^{i,2} = 3$, $r_0^{i,1} = r_0^{i,2} = 1/2$, and $d_x^{i,k} = 0, \forall(x, i, k)$.[14] Their $T_t^{i,k}$, however, are set differently such as $T_t^{i,1} = 7$, $T_t^{i,2} = 11$. The first row of each example depicts the expected arrival times of $n_r^{i,k}$ to-be-arriving flits based on Condition 2, where the first two to-be-arriving flits arrive sequentially and the tail flit arrive after $T_t^{i,k}$. Then, the following three rows represent

[13]The last OQ in the routing path is the OQ of the second last router, i.e., the router just before the destination router.

[14]To extend the example for non-zero $d_x^{i,k}$, we just need to shift (to the right) the transmission timing of each flit through link $l_x^{i,k}$ by $d_x^{i,k}$.

when each flit is transmitted through a set of links. At the bottom row, two terms of Eq. (2) are visualized as green boxes for the first term and a red box for the second term. In the case when $\frac{n_q^{i,k}+n_r^{i,k}-1}{\rho_m^{i,k}}$ is greater than $T_t^{i,k}$ like in Fig. 8(a), all the flits are expected to be sequentially transmitted through the lowest-rate output link during $\frac{n_q^{i,k}+n_r^{i,k}-1}{\rho_m^{i,k}}$ slots. Otherwise, there exists a gap between the transmission of the second last flit and the tail flit at the lowest-rate output link as depicted in Fig. 8(b) where the time it takes to transmit flits excluding the tail flit and wait for the arrival of the tail flit at the OQ with the lowest-rate link is equal to $T_t^{i,k}$. To take account of these situations, we have introduced a max operation in Eq. (2).

The proposed packet discarding mechanism calculates $\underline{\mathrm{D}}_r^{i,k}$ for every queued packet in an OQ. Therefore, its computational complexity is given as $O(n)$, where n is the number of packets in an OQ. Note that calculation of $\underline{\mathrm{D}}_r^{i,k}$ by Eq. (2) requires negligible computation load, because (i) the first term in Eq. (2) can be calculated only once at a new packet's arrival, and (ii) the second term needs occasional updates of $n_q^{i,k}$, $n_r^{i,k}$, $T_t^{i,k}$ along with a simple comparison for the max operation.

B. EARLIEST DEADLINE FIRST FLIT SCHEDULING ALGORITHM

The first flit scheduling algorithm to propose is *f*EDF (*f*lit-based *E*arliest *D*eadline *F*irst), which is inspired by the classical EDF algorithm for scheduling CPU jobs [33]. The classical EDF assigns higher priority to the job with an earlier deadline, and is known to be optimal in minimizing maximum lateness where the lateness of a job is defined as its completion time *minus* its deadline [34]. Note that given a set of *schedulable* jobs,[15] EDF guarantees to keep the schedulability while achieving the optimality in the sense of lateness.

When adopting EDF, the proposed *f*EDF redefines the notion of 'deadline' to properly fit into the CT switching context in two different ways, according to which two types of *f*EDF are developed, denoted by *f*EDF-dst and *f*EDF-hop. First, *f*EDF-dst considers the *to-destination* deadline when scheduling the flits of packet $P^{i,k}$, which is defined as the remaining time until the packet's required arrival time at the destination, calculated as $(T_d^{i,k} + D_{req}^i - t)$ where t denotes the current time. Next, *f*EDF-hop deals with the *per-hop* deadline of packet $P^{i,k}$ at an OQ, which is defined as the to-destination deadline *divided by* $l_{path}^{i,k}$, to capture the impact of the number of the remaining links to the destination. Then at an OQ, the *f*EDF (either *f*EDF-dst or *f*EDF-hop) algorithm calculates the deadline of every existing packet,[16] and the queued flit belonging to the packet with the minimal value (i.e., the earliest deadline) is chosen to transmit with the transmission rate equal to the output link rate. As a result, the computational complexity of *f*EDF is given as $O(n) + O(n) = O(n)$, which includes calculating the deadlines of n queued packets and comparing them to find the smallest deadline.

Intuitively, *f*EDF-dst is a straightforward extension of EDF by interpreting the deadline to arrive at the destination router as the job deadline at a CPU and the remaining routers (including the current one) collectively as the CPU. On the other hand, *f*EDF-hop considers the per-hop deadline instead, under the perspective that flit scheduling is conducted independently at each router and thus each OQ should consider its own portion of the deadline when adopting EDF. Therefore, it is certain that the two algorithms would perform similar to each other when l_{path}^i does not vary much among the existing packets in an OQ.

Fig. 9 depicts the difference between *f*EDF-dst and *f*EDF-hop, where two packets $P^{1,a}$ and $P^{2,b}$ from different sessions share the link $L_1^{1,a}(= L_1^{2,b})$ in their remaining paths with $l_{path}^{1,a} = 2$ and $l_{path}^{2,b} = 4$, as shown in Fig. 9(b). Note that $d_x^{i,k} = 0$ and $\rho_x^{i,k} = 1$ for all links, $n_q^{i,k} = 2$ and $n_r^{i,k} = 0$ for both packets as shown in Fig. 9(a). The to-destination deadlines of $P^{1,a}$ and $P^{2,b}$ are given as 5 and 6, respectively. When *f*EDF-dst is adopted, the upper-leftmost router transmits the flits of $P^{1,a}$ first and those of $P^{2,b}$ later since $P^{1,a}$ has the minimal to-destination deadline, as shown in Fig. 9(c). As a result, the latency of $P^{2,b}$ (which is 7) exceeds its given to-destination deadline (which is 6).[17] In case of *f*EDF-hop as illustrated in Fig. 9(d), however, $P^{2,b}$ arrives at the destination within the deadline because it gets higher priority than $P^{1,a}$. Moreover, $P^{1,a}$ also satisfies its delay constraint.

Discussion: Intuitively, *f*EDF-hop should outperform *f*EDF-dst when l_{path}^i varies significantly in an OQ, thanks to its consideration to the per-hop share of the deadline. Nevertheless, it is not straightforward to expect that the variant of *f*EDF would achieve the best performance in both end-to-end latency and PDR, since the CT switching network environment cannot always guarantee the schedulability requirement of EDF due to the random nature of network congestion. That is, when PDR is high but not 100% (which is quite common in a network), the desirable properties of EDF would be impaired to some extent. The impact of imperfect PDR on *f*EDF will be further investigated and discussed in Section V via extensive simulations.

C. SHORTEST PROCESSING-TIME FIRST FLIT SCHEDULING ALGORITHM

The second flit scheduling algorithm we propose is *f*SPF (*f*lit-based *S*hortest *P*rocessing-time *F*irst), which is inspired by the SJF algorithm for scheduling CPU jobs [33]. SJF assigns higher priority to the job having smaller CPU

[15] A set of jobs is schedulable if there exist one or more solutions to schedule all the jobs within their deadlines.

[16] An existing packet implies the one whose head flit has arrived earlier but tail flit has not yet been transmitted to the next router.

[17] If the packet discarding mechanism is adopted, $P^{2,b}$ should be discarded at the beginning of slot 3, because $\underline{\mathrm{D}}_r^{2,b} = 5$. To show the difference between *f*EDF-dst and *f*EDF-hop more clearly, we intentionally ignored the discarding mechanism.

Packet	$n_q^{i,k}$	$l_{path}^{i,k}$	Deadline	
			To-dst.	Per-hop
$P^{1,a}$	2	2	5	2.5
$P^{2,b}$	2	4	6	1.5

(a)

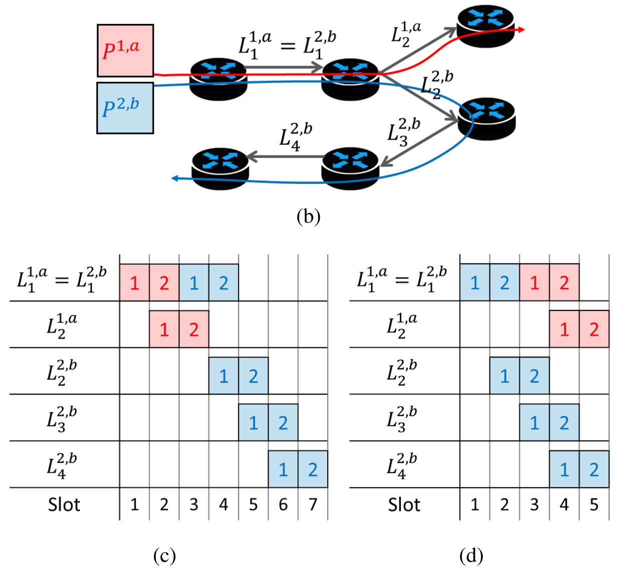


FIGURE 9. *f*EDF-dst vs. *f*EDF-hop. (a) Output queue status. (b) Network topology. (c) *f*EDF-dst. (d) *f*EDF-hop.

burst, and is known to be optimal in minimizing the average waiting time. Note that the waiting time can be interpreted as the queueing delay in the CT switching network.

Similar to *f*EDF, *f*SPF re-defines the notion of 'job' in SJF as *processing time* and does so in two different ways so that two types of *f*SPF are designed, denoted by *f*SPF-dst and *f*SPF-hop. First, *f*SPF-dst defines the *to-destination* processing time as the expected time remaining (based on Conditions 1 and 2) until the tail flit of a packet arrives at the destination, which is equal to $D_r^{i,k}$ for packet $P^{i,k}$. On the other hand, *f*SPF-hop considers the *per-hop* processing time T_p defined as the minimal expected remaining time (based on Conditions 1 and 2) until the tail flit of a packet is transmitted by the current router. Then, in an OQ, either of the two *f*SPF algorithms runs to identify the packet having the shortest (per-packet) processing time, and transmits the oldest queued flit of the packet with the output link rate. Note that the processing time of each packet is re-calculated once the chosen flit is transmitted.

Based on Condition 1, T_p of packet $P^{i,k}$, denoted by $T_p^{i,k}$, can be calculated using Eq. (2) by interpreting the next router as a virtual destination. That is, by setting $l_{path}^{i,k} = 1$, $\rho_m^{i,k} = \rho_1^{i,k}$ in Eq. (2), $T_p^{i,k}$ is obtained as

$$T_p^{i,k} = \frac{1}{\rho_1^{i,k}} + d_1^{i,k} + \max\left(\frac{n_q^{i,k} + n_r^{i,k} - 1}{\rho_1^{i,k}}, T_t^{i,k}\right). \quad (3)$$

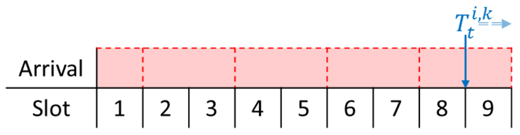


FIGURE 10. An example checking $T_t^{i,k}$.

Then, the computational complexity of *f*SPF becomes $O(n)$ for n queued packets in the given OQ, which includes calculating the processing times of n queued packets and comparing them to find the shortest processing time.

Using $T_p^{i,k}$, $T_t^{i,k}$ can be also calculated and informed to the next router in the following steps.

1) For each packet, $T_p^{i,k}$ is calculated by using $T_t^{i,k}$ reported by the upstream router.
2) The flit to be transmitted is determined by the proposed scheduling algorithm (*f*SPF or *f*EDF).
3) $T_t^{i,k} = T_p^{i,k} - 1/\rho_1^{i,k}$ is embedded in the transmitted flit.

In flit scheduling, a series of incoming flits belonging to $P^{i,k}$ can be interrupted by another packet $P^{i',k'}$, e.g., $P^{i',k'}$ suddenly arrives at an upstream router and gets higher priority than $P^{i,k}$ while $P^{i,k}$ was being transmitted. In that case, the current router cannot have an updated $T_t^{i,k}$ until the upstream router resumes transmission of the to-be-arriving flits of $P^{i,k}$. Therefore, the current router needs to check if the tail flit of $P^{i,k}$ can actually arrive within $T_t^{i,k}$ slots by comparing the previously advertised $T_t^{i,k}$ with the minimum possible time for receiving $n_r^{i,k}$ to-be-arriving flits, which is determined as $(n_r^{i,k} - 1)/r_0^{i,k} + 1$ by considering that at best the first to-be-arriving flit arrives at the next slot and other to-be-arriving flits arrive sequentially with the incoming rate of $r_0^{i,k}$. If $T_t^{i,k} < (n_r^{i,k} - 1)/r_0^{i,k} + 1$, the transmission of $P^{i,k}$ is judged to be postponed and $T_t^{i,k}$ is updated to $(n_r^{i,k} - 1)/r_0^{i,k} + 1$. For example, in Fig. 10, if $T_t^{i,k} = 8$, $n_r^{i,k} = 5$, and $r_0^{i,k} = 1/2$, the minimum time for receiving to-be-arriving flits is 9 which is greater than $T_t^{i,k}$, and thus $T_t^{i,k}$ is updated to 9.

Discussion: The classical SJF cannot guarantee that given jobs complete within their deadlines as EDF does. Nevertheless, we expect *f*SPF can provide higher PDR than *f*EDF due to the uncertainty at the downstream routers described as follows. If a packet is discarded at a router due to congestion, all the resources consumed so far at the upstream routers to transmit the packet end up with being wasted. In such a case, *f*EDF would suffer from its strategy to assign higher priority to the packet with an earlier deadline, because such a packet has a higher chance to violate its requirement. *f*SPF, however, prefers to transmit the packet with minimal processing time, and hence the probability that the packet would be discarded is smaller than *f*EDF. Moreover, even if the packet is discarded, the amount of wasted resources would be comparatively small thanks to the nature of SJF. In Section V, the superiority of *f*SPF will be demonstrated via extensive simulations.

TABLE 2. Configuration of simulations.

	Sim-1	Sim-2	Sim-3
U_{avg} (%)	40	20, 30, 40, 50, 60	20, 30, 40, 50, 60, 70
Calculation of U_{avg}	based on SF switching	based on SF switching	based on CT switching
Number of sessions	150	600	600
Packet size, $s_p^{i,k}$ (bytes)	fixed (300, 600, 900, 1200, or 1500)	mixed (600, 900, 1200, and 1500)	mixed (600, 900, 1200, and 1500)
Flit size, s_f (bytes)	variable	304	304
Delay requirement, D_{req}^i (ms)	10	10	10, 7

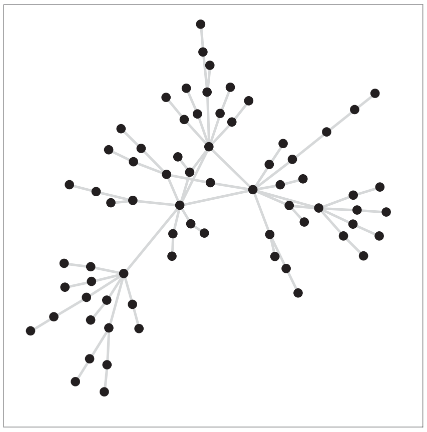


FIGURE 11. Network topology used in the simulation.

V. PERFORMANCE EVALUATION

We evaluate the performance of the two proposed scheduling algorithms in terms of the average end-to-end latency and PDR, via extensive simulation. To verify the efficacy of CT-based flit scheduling for a NZL service, we compare the proposed methods with iSLIP [25] and two variants of FIFO (First-In First-Out) scheduling, *FIFO-SF* (FIFO with SF switching only) and *FIFO-CT* (FIFO with CT switching only). Note that iSLIP, FIFO-SF, and FIFO-CT discard a packet if its accumulated latency becomes greater than the delay requirement. Also note that in all schemes, the average end-to-end latency is measured using non-discarded packets.

In the simulation, we use a network topology derived from a real world network, RedClara, which connects Latin America's academic networks [35], while scaling it down to fit into a nation-wide network. The number of routers is 71, and 30 of them are source or destination routers. There exist $N_l = 148$ links with various link rates in the range from 2.5 to 20 Mbps, and we set $L_\rho = 20$ Mbps. The average per-link propagation delay is set to 0.4 ms. In addition, we assign the same routing path to the packets belonging to a session. Fig. 11 illustrates the employed topology where vertices and edges represent routers and links, respectively.

To represent the traffic load on the network, we define the average resource usage U_{avg} as[18]

$$U_{avg} = \frac{1}{N_l} \sum_{1 \le j \le N_l} \left[\frac{\sum_{S^i \text{ crossing link } j} (\lambda^i/\tau) \cdot l^i_{PHY}}{\rho_j} \right], \tag{4}$$

[18]Eq. (4) is formulated under the assumption that the packets belonging to the same session are forwarded through the common routing path.

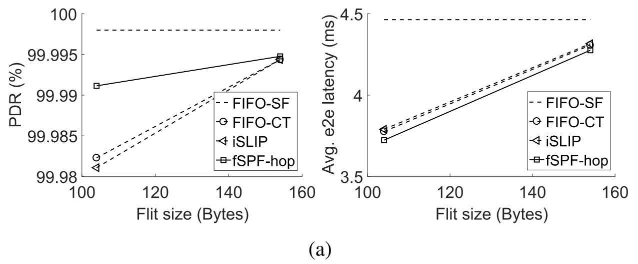
where λ^i/τ (packets/sec) is the arrival rate of session S^i, ρ_j (bps) is the link rate of link j, and l^i_{PHY} (bits) denotes the average packet size of session S^i in the PHY layer such as $l^i_{PHY} = 8 \cdot (E[s_p^{i,k}] + s_o)$ for SF switching and $l^i_{PHY} = 8 \cdot (\lceil E[s_p^{i,k}]/(s_f - s_h) \rceil \cdot (s_f + s_o))$ for CT switching. Hence, $\sum_{S^i \text{ crossing link } j} (\lambda^i/\tau) \cdot l^i_{PHY}/\rho_j$ implies the average traffic rate passing through link j *divided by* link j's rate, which measures the average utilization of the link. Accordingly, U_{avg} becomes the average per-link utilization over all N_l links. In the sequel, we vary U_{avg} to test the performance under diverse network loads, and $U_{avg}(\%)$ will imply $U_{avg} \times 100$.

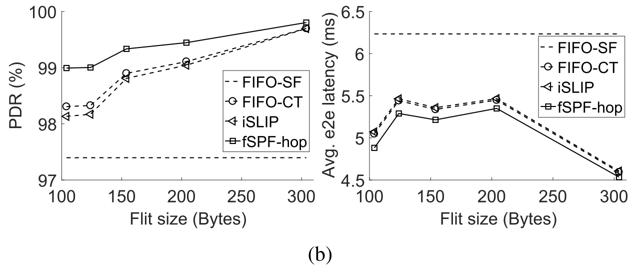
We carried out three types of simulation, *Sim-1* (in Section V-A) to find the optimal flit size, *Sim-2* (in Section V-B) to verify the benefit of CT switching against SF switching, and *Sim-3* (in Sections V-C, V-D, V-E) to compare the performance of the proposed algorithms with FIFO and iSLIP. In Sim-1, we set the size of all packets to either 300, 600, 900, 1200, or 1500 (bytes), with 150 sessions and $U_{avg} = 40\%$ (which is close to the median U_{avg} of Sim-2 and Sim-3). In Sim-2 and Sim-3, U_{avg} is varied as 20-60% and 20-70%, respectively, and a total of 600 sessions are split into four groups where each group consists of 150 sessions and transmits fixed-size packets (either 600, 900, 1200, or 1500 bytes, respectively). In Sim-1 and Sim-2, U_{avg} is measured from the viewpoint of SF switching,[19] while U_{avg} in Sim-3 is measured from the viewpoint of CT switching. Regarding packet arrivals, Poisson arrival process is applied to each session with an arrival rate λ^i where each λ^i is set to proportional to given U_{avg}. Regarding the delay requirement, Sim-1 and Sim-2 consider 10 ms, whereas Sim-3 compares two values, 10 ms and 7 ms to investigate the impact of the deadline on scheduling performance. Finally, each simulation was conducted for an enough simulation time to observe converged performance. Table 2 summarizes the configuration of the simulations.

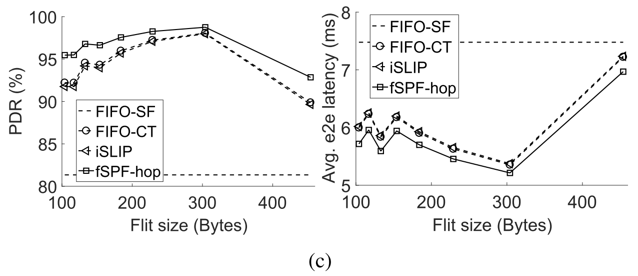
A. FLIT SIZE OPTIMIZATION

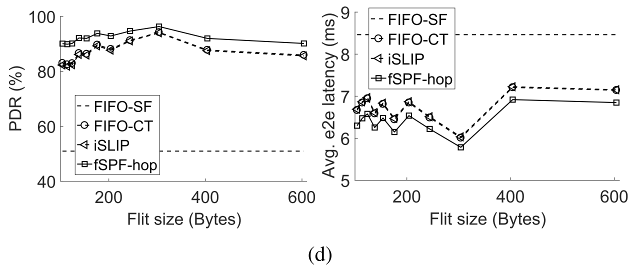
In this section, we try to determine the optimal flit size for each CT scheduling algorithm with varying packet size as $s_p^{i,k} = 300, 600, 900, 1200, 1500$ (bytes). For each $s_p^{i,k}$, we consider the values of s_f (in bytes) that correspond to the local minima of the flitization overhead (as described in Theorem 1), in the range of $104 \le s_f \le \lceil s_p^{i,k}/2 \rceil + 4$. Note

[19]In the same environment, U_{avg} of CT switching is around 10% larger than that of SF switching due to the flitization overhead, as will be shown in Section V-A.


(a)


(b)


(c)


(d)

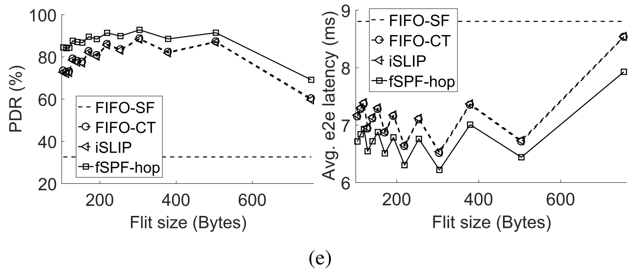

(e)

FIGURE 12. PDR and average end-to-end latency with varying flit size. (a) $s_p^{i,k} = 300$ bytes. (b) $s_p^{i,k} = 600$ bytes. (c) $s_p^{i,k} = 900$ bytes. (d) $s_p^{i,k} = 1200$ bytes. (e) $s_p^{i,k} = 1500$ bytes.

that $64 \leq s_f < 104$ is excluded since it leads to inferior performance both in latency and PDR due to huge flitization overhead.

Fig. 12 shows the simulation results of FIFO-SF, FIFO-CT, iSLIP, and *f*SPF-hop.[20] Except when $s_p^{i,k} = 300$, $s_f = 304$ achieves minimal average end-to-end latency and maximal PDR at every algorithm. With decreasing s_f, transmission

[20]Other three proposed algorithms are intentionally omitted due to their similarity to *f*SPF-hop.

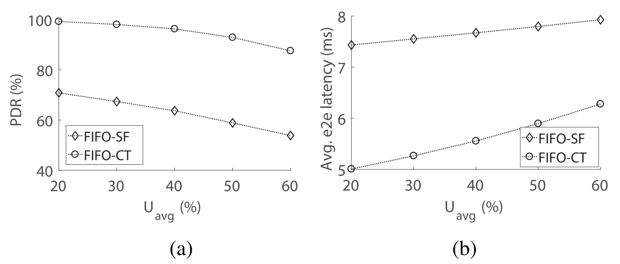

(a) (b)

FIGURE 13. Scheduling performance: SF switching vs. CT switching. (a) PDR. (b) Average end-to-end latency.

delay reduction by CT switching gets more prominent in general, but the flitization overhead sharply increases as well in the small s_f regime as shown earlier in Fig. 4 of Section III. As a result, the minimal latency is achieved at a moderate sized $s_f = 304$. With $s_f = 304$, the flitization overhead is measured as 1.078, 1.102, 1.114, 1.122 for $s_p^{i,k} = 600, 900, 1200, 1500$, respectively. When $s_p^{i,k} = 300$, however, PDR of CT switching gets even smaller than that of SF switching, from which we can conclude that *it is inappropriate to apply CT switching to such small size packets*. Motivated by the above observations, we consider in the sequel $s_p^{i,k} = 600, 900, 1200, 1500$ and $s_f = 304$.

B. SF SWITCHING VS. CT SWITCHING

In our simulation setup, CT switching incurs an additional overhead due to the flit headers and zero padding at the tail flit, resulting in the increase of per-packet bytes by 10.4% on average. Nevertheless, Fig. 13 shows that CT switching significantly outperforms SF switching in both PDR and latency, thanks to the sharp decrease in the transmission delay by CT switching. While SF switching never reaches the high PDR regime (e.g., 90%), CT switching can achieve 90% or higher PDR only in the limited cases of $U_{avg} \leq 50\%$, showing that CT-based packet forwarding alone cannot guarantee good performance, leading to the need of more sophisticated flit scheduling than FIFO.

C. SCHEDULING PERFORMANCE WITH VARYING NETWORK LOAD

Fig. 14 presents the performance comparison between the four proposed algorithms and two reference schemes, FIFO and iSLIP, when the delay requirement is 10 ms for every session. As seen, FIFO and iSLIP perform most poorly among all but very similarly to each other in both PDR and average end-to-end latency. While the poor performance of FIFO is not a surprise, iSLIP performs not well partly because head-of-line blocking and starvation at input queues (which are what iSLIP deals with) never occur in our model. In the mean time, compared to FIFO, *f*SPF-dst most enhances PDR by 11.52% while *f*SPF-hop most enhances average end-to-end latency by 13.86%.

Fig. 15 shows scheduling performance when $D_{req}^i = 7$ ms, $\forall i$, which clearly differs from Fig. 14. Compared to the 10 ms case, the achieved average end-to-end latency tends to be smaller whereas the achieved PDR is degraded,

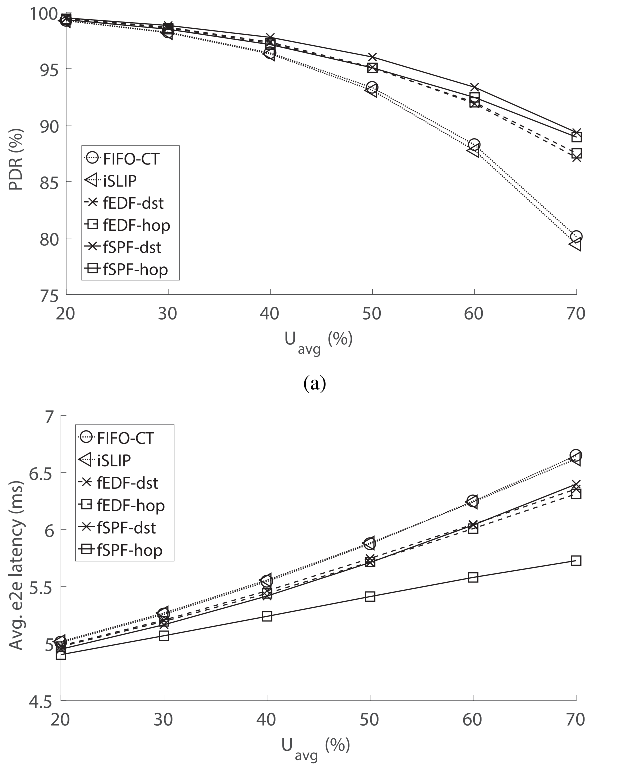


FIGURE 14. Scheduling performance: With varying U_{avg} and D^{i}_{req} = 10ms. (a) PDR. (b) Average end-to-end latency.

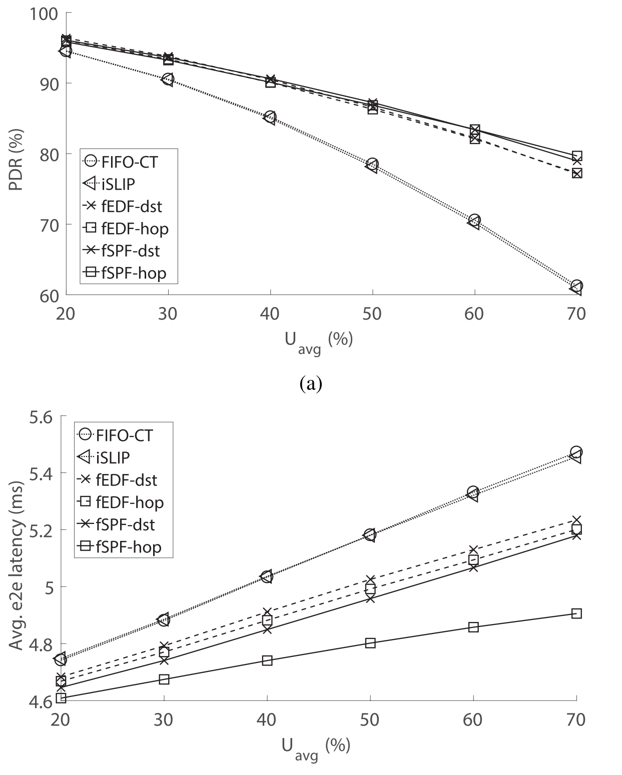


FIGURE 15. Scheduling performance: With varying U_{avg} and D^{i}_{req} = 7ms. (a) PDR. (b) Average end-to-end latency.

due to the tightened delay requirement. In terms of the performance enhancement by the proposed algorithms against FIFO, *f* SPF-hop achieves the largest enhancement in both PDR and average end-to-end latency such as 30.11% PDR enhancement and 10.33% latency enhancement. Note that PDR enhancement is much greater with 7 ms (30.11% with 7 ms vs. 11.52% with 10 ms) while latency enhancement is better with 10 ms (13.86% with 10 ms vs. 10.33% with 7 ms). Considering that enhancing PDR gets more important once the delay requirement is met, the efficacy of the proposed schedulers seems to become more outstanding with tighter delay requirement.

Another observation from Figs. 14 and 15 is that *f* SPF always performs better than *f* EDF. In fact, the superiority of *f* SPF against *f* EDF is counter-intuitive since by design, *f* EDF aims at enhancing PDR while *f* SPF focuses on latency reduction. The observed phenomenon can be explained as follows. *f* EDF prefers to schedule the packets closer to their deadlines, thus allocating more resources to them. Since such packets are more likely to violate the delay requirement, the already-allocated resources may end up with being wasted once they are discarded later on. As a result, *f* EDF may not achieve its intended goal of good PDR performance.

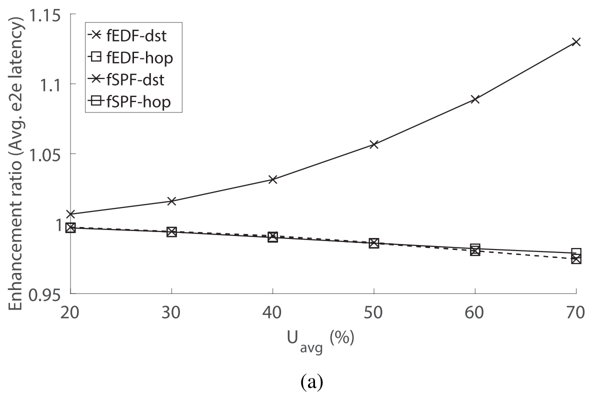
D. EFFECT OF PACKET DISCARDING

Remind that in our packet discarding mechanism, any packet expected to eventually violate the deadline is discarded proactively, rather than waiting until its violation. To quantify the impact of packet discarding on CT switching, we define a new metric called the *enhancement ratio* as: the PDR (or average end-to-end latency) achieved by the proposed packet discarding mechanism *divided by* the PDR (or average end-to-end latency) achieved by a naive method that discards a packet only when its accumulated delay exceeds the requirement. Therefore, the larger the ratio in PDR becomes, the more enhanced the PDR performance is; whereas the smaller the ratio in average end-to-end latency becomes, the more enhanced the latency performance is. Note that thus-defined enhancement ratio indicates how much our packet discarding can improve latency and PDR compared to the case it's not applied.

Fig. 16 presents the enhancement ratio of the four proposed algorithms, and Table 3 compares the proposed discarding mechanism with the naive method in terms of the ratio of the discarded packets (in %) for various U_{avg}. It is clear that

TABLE 3. Ratio of discarded packets (in %) for U_{avg} = 20, 30, 40, 50, 60, 70 (%).

	Proposed method	Naive method
*f*EDF-dst	0.5, 1.3, 2.7, 4.9, 8.1, 12.9	0.6, 1.6, 3.4, 6.5, 11.2, 18.4
*f*EDF-hop	0.6, 1.4, 2.8, 4.9, 8.0, 12.5	0.7, 1.7, 3.6, 6.8, 11.4, 18.4
*f*SPF-dst	0.5, 1.2, 2.2, 4.0, 6.6, 10.6	0.9, 2.2, 4.2, 7.3, 10.9, 15.6
*f*SPF-hop	0.7, 1.5, 2.9, 4.9, 7.5, 11.1	0.7, 1.7, 3.4, 6.0, 9.5, 14.1



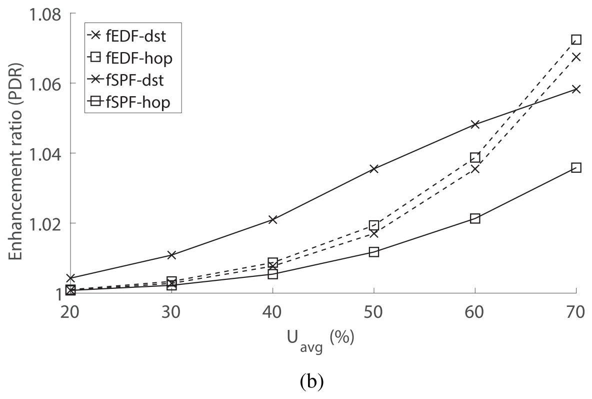
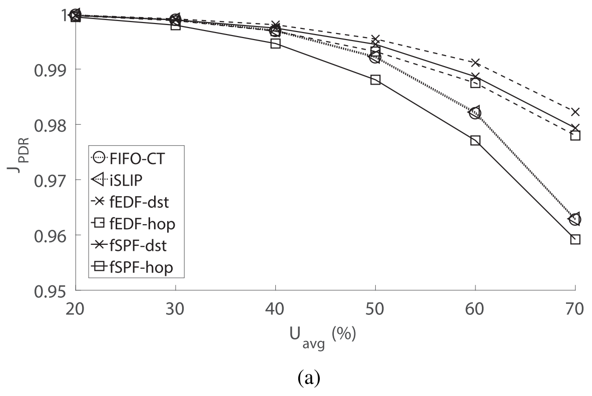


FIGURE 16. Scheduling performance enhancement by the packet discarding mechanism. (a) Average end-to-end latency. (b) PDR.



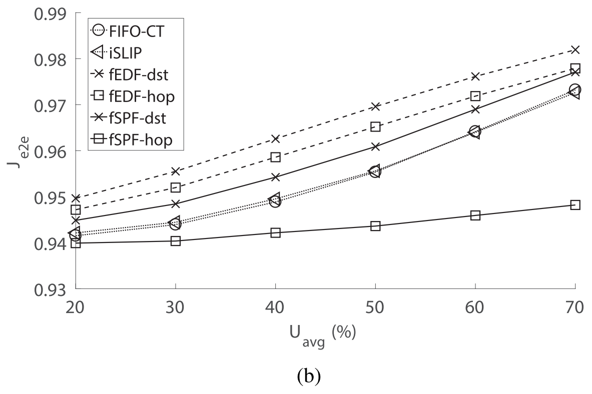


FIGURE 17. Achieved fairness among sessions. (a) PDR. (b) Average end-to-end latency.

the packet discarding scheme enhances the performance of all the algorithms but *f* SPF-dst, in both PDR and latency. On the other hand, *f* SPF-dst achieves degraded latency but enhanced PDR, achieving the largest enhancement in PDR among the proposed algorithms for $U_{avg} \leq 60\%$ and the third best for larger U_{avg}. This can be interpreted as (i) *f* SPF-dst secures more network resources by more proactively discarding soon-to-violate packets, using which the packets on the verge of violation can reach their destinations within the deadline, and (ii) such "resurrected" packets tend to achieve large (but less than the deadline) end-to-end latency resulting in the increased average end-to-end latency. Moreover, Table 3 shows that such proactive discarding eventually results in less discarded packets than the naive method, which confirms the efficacy of the proposed method.

E. FAIRNESS BETWEEN SESSIONS WITH VARYING NETWORK LOAD

To show how much fairness between sessions is achieved by the proposed scheduling algorithms, we define the fairness in PDR (denoted by J_{PDR}) and the fairness in the average end-to-end latency (denoted by J_{e2e}) such as

$$J_{PDR}(S^1, S^2, \cdots, S^{ns}) := \frac{\left(\sum_{i=1}^{ns} p^i\right)^2}{n_S \cdot \sum_{i=1}^{ns} {p^i}^2}, \quad (5)$$

$$J_{e2e}(S^1, S^2, \cdots, S^{ns}) := \frac{\left(\sum_{i=1}^{ns} l^i\right)^2}{n_S \cdot \sum_{i=1}^{ns} {l^i}^2}, \quad (6)$$

where n_S is the number of sessions (600 in this simulation), p^i is the PDR of session S^i, and l^i the average end-to-end latency of session S^i. Note that Eqs. (5) and (6) are slightly modified from the Jain's fairness index [36] by replacing throughput with PDR or latency.

Fig. 17 presents that FIFO and iSLIP performs similarly to each other. In addition, regarding *f* EDF and *f* SPF, to-destination schemes are more fair than per-hop schemes both in PDR and latency. In all cases, as U_{avg} grows, J_{e2e} increases gradually while J_{PDR} drops fast. Nevertheless, all the algorithms seem very fair since they achieve $J_{PDR} > 0.95$ and $J_{e2e} > 0.94$.

VI. CONCLUSION AND FUTURE WORK

In this paper, we considered cut-through switching with flitization of packets to significantly reduce the end-to-end latency in general-purpose data communication networks. Accordingly, we proposed a packet discarding mechanism to shed likely-delay-violated packets in advance, and two flit scheduling algorithms to achieve good performance in end-to-end latency and reliability. Through extensive simulations, we confirmed the performance enhancement by the two proposed schemes, and revealed that the SJF-inspired algorithm achieves better performance than the EDF-inspired algorithm.

In future, we would like to develop more advanced schemes to effectively reduce the end-to-end latency with high reliability, such as session admission control and scheduling-aware flit routing.

APPENDIX A PROOF OF THEOREM 1

Assume that a packet with size $\theta \leq 1,500$ (bytes) is split into n flits, where each flit consists of 4 (bytes) of flit header and y (bytes) of flit payload, i.e., $s_f = 4 + y$. Due to the minimal flit size discussed in Section III-B, we have

$$s_f = 4 + y \geq 64 \quad \Rightarrow \quad y \geq 60. \tag{7}$$

In addition, since n copies of flit payloads include θ,

$$\theta \leq ny \quad \Rightarrow \quad \theta/n \leq y \quad \Rightarrow \quad \lceil \theta/n \rceil \leq y, \quad y \in \mathbb{N} \tag{8}$$

Because Eq. (7) is a necessary condition and Eq. (8) specifies the range of all possible values of y, we have $\lceil \theta/n \rceil \geq 60$.

In the mean time, we can also express θ as $\theta = n{\cdot}a{+}b,\ a \geq 0,\ 0 \leq b < n$, for $\theta, n, a, b \in \mathbb{Z}^+ := \{0, 1, 2, \ldots\}$, based on which we notice that

$$\left\lceil \frac{\theta}{n} \right\rceil = \left\lceil a + \frac{b}{n} \right\rceil = \begin{cases} a, & \text{if } b = 0, \\ a+1, & \text{otherwise } (0 < b < n). \end{cases} \tag{9}$$

Combining this with $\lceil \theta/n \rceil \geq 60$, we obtain $a \geq 59$. Then,

$$59n + b \leq na + b = \theta \leq 1500 \Rightarrow n \leq 1500/59 \Rightarrow n \leq 25.$$

Moreover, $b < n$ with $a \geq 59$ and $n \leq 25$ leads to $b < a$.

In the sequel, we first prove three lemmas, and then using the lemmas we prove the local minimality at $s_f = \lceil \theta/n \rceil + s_h, n = 2, 3, \cdots$. In addition, we will denote by δ_n a set of integers in $\left[0, \lceil \frac{\theta}{n-1} \rceil - \lceil \frac{\theta}{n} \rceil\right)$, i.e., $\delta_n \in \mathbb{Z}^+, 0 \leq \delta_n < \lceil \frac{\theta}{n-1} \rceil - \lceil \frac{\theta}{n} \rceil$. Accordingly, we have $\lceil \frac{\theta}{n} \rceil \leq \delta_n + \lceil \frac{\theta}{n} \rceil < \lceil \frac{\theta}{n-1} \rceil$. This way, we will be able to express whatever integers in $\left[\lceil \frac{\theta}{n} \rceil, \lceil \frac{\theta}{n-1} \rceil\right)$ while varying δ_n.

Lemma 1: We have $-1 < \frac{-n\delta_n}{a+\delta_n} \leq 0$ *for* $b = 0$, *and* $-1 < \frac{b-n\delta_n-n}{a+\delta_n+1} < 0$ *for* $0 < b < n$.

Proof: We first prove the case for $b = 0$. When $b = 0$, we have $a > 0$ (since $\theta \neq 0$) and thus $\frac{-n\delta_n}{a+\delta_n} \leq 0$. In addition,

$$\begin{aligned} \delta_n &< \left\lceil \frac{\theta}{n-1} \right\rceil - \left\lceil \frac{\theta}{n} \right\rceil = \left\lceil \frac{na}{n-1} \right\rceil - a \\ &= \left\lceil a + \frac{a}{n-1} \right\rceil - a = \left\lceil \frac{a}{n-1} \right\rceil, \end{aligned} \tag{10}$$

where the last equality holds since a is an integer. Because δ_n is an integer, Eq. (10) implies $\delta_n < \frac{a}{n-1}$, which directly leads to $\frac{-n\delta_n}{a+\delta_n} > -1$.

Next, we prove the case for $0 < b < n$. When $0 < b < n$, it is straightforward that $\frac{b-n\delta_n-n}{a+\delta_n+1} < 0$ since $b - n < 0$. In addition,

$$\begin{aligned} \delta_n &< \left\lceil \frac{\theta}{n-1} \right\rceil - \left\lceil \frac{\theta}{n} \right\rceil = \left\lceil \frac{na+b}{n-1} \right\rceil - (a+1) \\ &= \left\lceil a + \frac{a+b}{n-1} \right\rceil - a - 1 = \left\lceil \frac{a+b}{n-1} \right\rceil - 1, \end{aligned} \tag{11}$$

where the first equality is due to Eq. (9) and the last equality holds since a is an integer. Because δ_n is an integer, Eq. (11) implies $\delta_n < \frac{a+b}{n-1} - 1$, from which $\frac{b-n\delta_n-n}{a+\delta_n+1} > -1$ is obtained. This completes the proof. ■

Lemma 2: We have $\lceil \theta/(n+1) \rceil \leq \lceil \theta/n \rceil - 1$.

Proof: We prove the claim by contradiction. Assume $\lceil \theta/(n+1) \rceil > \lceil \theta/n \rceil - 1$. When $b = 0$, this leads to

$$\left\lceil \frac{na}{n+1} \right\rceil = \left\lceil a - \frac{a}{n+1} \right\rceil > a - 1 \Rightarrow \left\lceil \frac{-a}{n+1} \right\rceil > -1, \tag{12}$$

and when $0 < b < n$, it leads to

$$\left\lceil \frac{na+b}{n+1} \right\rceil = \left\lceil a + \frac{b-a}{n+1} \right\rceil > a \Rightarrow \left\lceil \frac{b-a}{n+1} \right\rceil > 0, \tag{13}$$

by the same logic that applied to Eq. (10). Eqs. (12) and (13), however, are contradictory to $a \geq 59$, $n \leq 25$, and $b < a$, which completes the proof. ■

Lemma 3: We have $\left\lceil \frac{\theta}{\lceil \theta/n \rceil - 1} \right\rceil = n+1$, *and* $\left\lceil \frac{\theta}{\lceil \theta/n \rceil + \delta_n} \right\rceil = n$.

Proof: By Lemma 2, $\lceil \theta/(n+1) \rceil \leq \lceil \theta/n \rceil - 1 < \lceil \theta/n \rceil$, and thus $\lceil \theta/n \rceil - 1$ can be expressed by $\lceil \theta/(n+1) \rceil + \delta_{n+1}$. By applying this to the first equality in the claim, we obtain the second equality with n replaced by $(n+1)$. Therefore, we just need to prove $\lceil \theta/(\lceil \theta/n \rceil + \delta_n) \rceil = n$.

By Eq. (9) and Lemma 1, we have

$$\begin{aligned} \left\lceil \frac{\theta}{\lceil \theta/n \rceil + \delta_n} \right\rceil &= \begin{cases} \left\lceil \dfrac{na}{a+\delta_n} \right\rceil, & \text{if } b = 0, \\ \left\lceil \dfrac{na+b}{a+\delta_n+1} \right\rceil, & \text{otherwise}, \end{cases} \\ &= \begin{cases} \left\lceil n + \dfrac{-n\delta_n}{a+\delta_n} \right\rceil, & \text{if } b = 0, \\ \left\lceil n + \dfrac{b-n\delta_n-n}{a+\delta_n+1} \right\rceil, & \text{otherwise}, \end{cases} \\ &= n, \end{aligned}$$

and accordingly,

$$\left\lceil \frac{\theta}{\lceil \theta/n \rceil - 1} \right\rceil = \left\lceil \frac{\theta}{\lceil \theta/(n+1) \rceil + \delta_{n+1}} \right\rceil = n+1, \tag{14}$$

which completes the proof. ■

Now, we prove the local minimality at $s_f = \lceil \theta/n \rceil + s_h, n = 2, 3, \cdots$. For the fixed packet size θ, Eq. (1) becomes

$$\frac{\lceil \frac{\theta}{s_f - s_h} \rceil \cdot \left(s_f + s_o\right)}{\theta + s_o}. \tag{15}$$

Since the denominator of Eq. (15) is constant, we focus on the numerator. By applying Lemma 3, we obtain the following: (1) when $s_f = \lceil \theta/n \rceil + s_h - 1$,

$$\begin{aligned} \left\lceil \frac{\theta}{s_f - s_h} \right\rceil \cdot \left(s_f + s_o\right) &= (n+1)\left(s_f + s_o\right) \\ &= (n+1)\left(\lceil \theta/n \rceil + s_h + s_o - 1\right), \end{aligned} \tag{16}$$

(2) when $s_f = \lceil \theta/n \rceil + s_h + \delta_n$,

$$\left\lceil \frac{\theta}{s_f - s_h} \right\rceil \cdot (s_f + s_o) = n\,(s_f + s_o) = n\,(\lceil \theta/n \rceil + s_h + s_o + \delta_n)\,. \tag{17}$$

Since Eq. (17) is monotonic increasing with $\delta_n \in [0, \lceil \theta/(n-1) \rceil - \lceil \theta/n \rceil - 1]$, the flitization overhead is minimal at $s_f = \lceil \theta/n \rceil + s_h$ (i.e., when $\delta_n = 0$) for $s_f \in [\lceil \theta/n \rceil + s_h, \lceil \theta/(n-1) \rceil + s_h - 1]$. In addition, we can show Eq. (17) at $\delta_n = 0$ is smaller than Eq. (16) such that

$$(n+1)\,(\lceil \theta/n \rceil + s_h + s_o - 1) - n\,(\lceil \theta/n \rceil + s_h + s_o) = \lceil \theta/n \rceil + s_h + s_o - (n+1) > 0,$$

by using the fact that $\lceil \theta/n \rceil \geq 60$, $s_h = 4$, $s_o = 42$, and $n \leq 25$. As a result, we conclude that the flitization overhead is minimized at $s_f = \lceil \theta/n \rceil + s_h$ for $s_f \in [\lceil \theta/n \rceil + s_h - 1, \lceil \theta/(n-1) \rceil + s_h - 1]$, for $n \geq 2$. Since the union of $[\lceil \theta/n \rceil + s_h - 1, \lceil \theta/(n-1) \rceil + s_h - 1]$, $n \geq 2$, includes every possible s_f, the above argument completes the proof.

REFERENCES

[1] A. Osseiran *et al.*, "Scenarios for 5G mobile and wireless communications: The vision of the METIS project," *IEEE Commun. Mag.*, vol. 52, no. 5, pp. 26–35, May 2014.

[2] G. P. Fettweis, "The tactile Internet: Applications and challenges," *IEEE Veh. Technol. Mag.*, vol. 9, no. 1, pp. 64–70, Mar. 2014.

[3] J. Y. Ngai and C. Setiz, "A framework for adaptive routing in multicomputer networks," *SIGARCH Comput. Archit. News*, vol. 19, no. 1, pp. 6–14, 1991.

[4] B. Grot, J. Hestness, S. Keckler, and O. Mutlu, "A comprehensive comparison between virtual cut-through and wormhole routers for cache coherent Network on-Chips," *SIGARCH Comput. Archit. News*, vol. 39, no. 3, pp. 401–412, 2011.

[5] R. Kapoor, G. Porter, M. Tewari, G. M. Voelker, and A. Vahdat, "Chronos: Predictable low latency for data center applications," in *Proc. ACM SoCC*, 2012, Art. no. 9.

[6] T. Szigeti, C. Hattingh, R. Barton, and K. Briley, *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks*, 2nd ed. London, U.K.: Pearson Education, 2013.

[7] J. W. Guck and W. Kellerer, "Achieving end-to-end real-time quality of service with software defined networking," in *Proc. IEEE CloudNet*, Oct. 2014, pp. 70–76.

[8] T. Wang, Z. Su, Y. Xia, and M. Hamdi, "Rethinking the data center networking: Architecture, network protocols, and resource sharing," *IEEE Access*, vol. 2, pp. 1481–1496, 2014.

[9] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," in *Proc. ACM SIGCOMM*, 2012, pp. 127–138.

[10] M. Alizadeh *et al.*, "pFabric: Minimal near-optimal datacenter transport," in *Proc. ACM SIGCOMM*, 2013, pp. 435–446.

[11] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM*, 2008, pp. 63–74.

[12] C. Guo *et al.*, "BCube: A high performance, server-centric network architecture for modular data centers," in *Proc. ACM SIGCOMM*, 2009, pp. 63–74.

[13] Y. Qian, Z. Lu, and Q. Dou, "QoS scheduling for NoCs: Strict priority queueing versus weighted round robin," in *Proc. IEEE ICCD*, Oct. 2010, pp. 52–59.

[14] M. ValadBeigi, F. Safaei, and B. Pourshirazi, "Application-aware virtual paths insertion for NOCs," *Microelectron. J.*, vol. 45, no. 4, pp. 454–462, 2014.

[15] M. Liu, M. Becker, M. Behnam, and T. Nolte, "Using non-preemptive regions and path modification to improve schedulability of real-time traffic over priority-based NoCs," *Real-Time Syst.*, vol. 53, no. 6, pp. 886–915, 2017.

[16] N. Su, K. Wang, X. Yu, H. Gu, Y. Guo, and J. Chen, "BARR: Congestion aware scheduling algorithm for network-on-chip router," *IEICE Electron. Express*, vol. 14, no. 3, 2017, Art. no. 20161247.

[17] N. J. Boden *et al.*, "Myrinet: A gigabit-per-second local area network," *IEEE Micro*, vol. 15, no. 1, pp. 29–36, Feb. 1995.

[18] M. D. Schroeder *et al.*, "Autonet: A high-speed, self-configuring local area network using point-to-point links," *IEEE J. Sel. Areas Commun.*, vol. 9, no. 8, pp. 1318–1335, Oct. 1991.

[19] A. Ademaj and H. Kopetz, "Time-triggered Ethernet and IEEE 1588 clock synchronization," in *Proc. IEEE ISPCS*, Oct. 2007, pp. 41–43.

[20] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Comput. Netw.*, vol. 3, no. 4, pp. 267–286, 1979.

[21] M. Ilyas and H. T. Mouftah, "Quasi cut-through: New hybrid switching technique for computer communication networks," *IEE Proc. E-Comput. Digit. Techn.*, vol. 131, no. 1, pp. 1–9, Jan. 1984.

[22] A. Abo-Taleb and H. Mouftah, "Delay analysis under a general cut-through switching technique in computer networks," *IEEE Trans. Commun.*, vol. 35, no. 3, pp. 356–359, Mar. 1987.

[23] J.-Y. Le Boudec, "The asynchronous transfer mode: A tutorial," *Comput. Netw. ISDN Syst.*, vol. 24, no. 4, pp. 279–309, 1992.

[24] A. Almulhem, F. El-Guibaly, and T. A. Gulliver, "Adaptive error correction for ATM communications using Reed–Solomon codes," in *Proc. IEEE SOUTHEASTCON*, Apr. 1996, pp. 227–230.

[25] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Trans. Netw.*, vol. 7, no. 2, pp. 188–201, Apr. 1999.

[26] *Cisco Nexus 5548P Switch Architecture*, Cisco, San Jose, CA, USA, Sep. 2010.

[27] X. Hao, H. Gu, B. Shu, D. Zeng, and Y. Li, "Performance evaluation of scheduling algorithms in network on chip," in *Proc. IEEE ICSPCC*, Sep. 2011, pp. 1–4.

[28] B. Nikolić and S. M. Petters, "EDF as an arbitration policy for wormhole-switched priority-preemptive NoCs—Myth or fact?" in *Proc. ACM ESWEEK*, 2014, pp. 1–10.

[29] A. Balasubramanian, B. Levine, and A. Venkataramani, "DTN Routing as a Resource Allocation Problem," in *Proc. ACM SIGCOMM*, 2007, pp. 373–384.

[30] U. Shevade, H. H. Song, L. Qiu, and Y. Zhang, "Incentive-aware routing in DTNs," in *Proc. IEEE ICNP*, Oct. 2008, pp. 238–247.

[31] Z.-S. Su. (May 1981). *A specification of the Internet Protocol (IP) Timestamp Option*. [Online]. Available: https://www.ietf.org/rfc/rfc781.txt

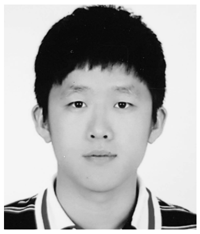
[32] *IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Standard 1588-2008 and IEEE Standard 1588-2002, IEEE Instrumentation and Measurement Society, 2008, pp. 1–300.

[33] A. Silberschatz, P. Galvin, and G. Gagne, *Operating System Concepts*, 8th ed. Hoboken, NJ, USA: Wiley, 2008.

[34] W. Horn, "Some simple scheduling algorithms," *Nav. Res. Logistics Quart.*, vol. 21, no. 1, pp. 177–185, 1974.

[35] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Recursively cautious congestion control," in *Proc. USENIX NSDI*, 2014, pp. 373–385.

[36] R. Jain, D.-M. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination," Eastern Res. Lab., Digit. Equip. Corp., Hudson, MA, USA, DEC Res. Rep., 1984.



KYUBO SHIN received the B.S. and M.S. degrees in electrical and computer engineering from the Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea, in 2013 and 2015, respectively, where he is currently pursuing the Ph.D. degree with the School of Electrical and Computer Engineering. His research interests include cognitive radio, super Wi-Fi, next-generation WLANs and cellular networks, and 5G communications.

SEOKWOO CHOI received the B.S. degree in electrical and computer engineering from the Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea, in 2017, where he is currently pursuing the M.S.–Ph.D. combined degrees with the School of Electrical and Computer Engineering. His research interests include 5G communications and low latency networking.

HYOIL KIM received the B.S. degree in electrical engineering from Seoul National University, Seoul, South Korea, in 1999, and the M.S. and Ph.D. degrees in electrical engineering: systems from the University of Michigan, in 2005 and 2010, respectively. Before joining UNIST, in 2011, he was a Postdoctoral Researcher with the IBM Thomas J. Watson Research Center, Hawthorne, NY, USA, from 2010 to 2011. He is currently an Associate Professor with the School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology (UNIST), Ulsan, South Korea. His research interests include wireless networking with an emphasis on cognitive radios (CR), mobile cloud, next-generation WLAN, and cellular networks, and 5G/6G communications. He serves in the Technical Program Committee for many renowned international conferences, such as the IEEE INFOCOM, ACM WiNTECH, the IEEE GLOBECOM, the IEEE PIMRC, the IEEE WCNC, and the IEEE VTC. He is an Associate Editor of *Computer Networks*, Elsevier, an Editor of the *Journal of Communications and Networks* (JCN), and a Guest Editor of the IEEE Transactions on Cognitive Communications and Networking (TCCN). He also served as a Guest Editor of *Ad Hoc Networks* for the special issue on Self-Organizing and Smart Protocols for Heterogeneous Ad hoc Networks.