IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 19, 2019, accepted January 2, 2020, date of publication January 13, 2020, date of current version January 23, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2966139

End-to-End Latency Prediction for General-Topology Cut-Through Switching Networks

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

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

This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) through the Korean Government (MSIT) under Grant 2015-0-00278 (research on near-zero latency network for 5g immersive service), in part by the National Research Foundation of Korea (NRF) through the Korean Government (MSIT) under Grant 2017R1C1B2011474, and in part by the Ulsan National Institute of Science and Technology (UNIST) under Grant 1.180065.01 and Grant 1.190035.01.

ABSTRACT Low latency networking is gaining attention to support futuristic network applications like the Tactile Internet with stringent end-to-end latency requirements. In realizing the vision, cut-through (CT) switching is believed to be a promising solution to significantly reduce the latency of today's store-and-forward switching, by splitting a packet into smaller chunks called *flits* and forwarding them concurrently through input and output ports of a switch. Nevertheless, the end-to-end latency performance of CT switching has not been well studied in heterogeneous networks, which hinders its adoption to general-topology networks with heterogeneous links. To fill the gap, this paper proposes an end-to-end latency prediction model in a heterogeneous CT switching network, where the major challenge comes from the fact that a packet's end-to-end latency relies on how and when its flits are forwarded at each switch while each flit is forwarded individually. As a result, traditional packet-based queueing models are not instantly applicable, and thus we construct a method to estimate per-hop queueing delay via M/G/c queueing approximation, based on which we predict end-to-end latency of a packet. Our extensive simulation results show that the proposed model achieves 3.98–6.05% 90th-percentile error in end-to-end latency prediction.

INDEX TERMS Computer networks, cut-through switching, end-to-end latency, M/G/c queueing model, queueing analysis, performance evaluation.

I. INTRODUCTION

With the advent of the 5G era, low latency has emerged as a key performance indicator (KPI) due to its essential role for soon-to-come immersive applications. For instance, the tactile Internet is expected to support remote haptic interactions requiring very low end-to-end latency of a few milliseconds [1]. In addition, virtual reality (VR) will require only 14 ms margin for computing and communication delay [2]. Moreover, industrial automation should forward packets end-to-end within 0.25–10 ms [3], while applications for robotics, telepresence, health care, and smart grid demand 1–10 ms latency [4]–[6].

The end-to-end latency in the network consists of transmission delay, queueing delay, processing delay, and propagation delay [7]. Among them, we focus on alleviating per-hop packet transmission and queueing delays, since processing delay is insignificant with today's computing-resource-rich high performance switches, and propagation delay is determined by the physical characteristics of the link thus not controllable. Especially, transmission delay does not scale well with the number of hops when store-and-forward (SF) switching is adopted, because a packet should be fully received before it can be forwarded to the next hop. Since SF switching is a de facto standard in most networks including the Internet, we should seek for an alternative switching mechanism for low-latency applications.

Cut-through (CT) switching, which was initially proposed in the late 1970s in [8] but has been used only in well-structured networks like data center networks (DCN) and Network-on-Chip (NoC), has recently been re-visited by [9] to reveal its potential to achieve low end-to-end latency

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

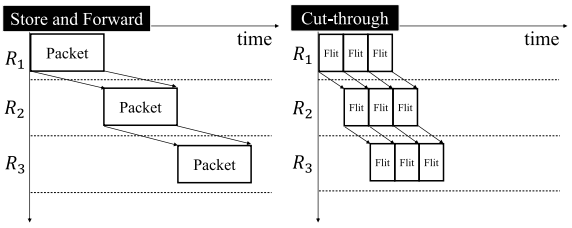


FIGURE 1. Store-and-forward vs. cut-through switching (R_1, R_2, R_3 denote router indices).

even in general-topology networks. In contrast to SF switching, CT switching can achieve minimal per-hop transmission delay by simultaneously forwarding a being-received packet via *flit-ization*, which is a process to split a packet into small chunks called *flits* (Fig. 1). More specifically, a packet is flitized at the source router (or at the gateway router of a CT switching domain) and each flit is forwarded individually (but in order).[1] The first flit of the packet should contain the full routing information so that a CT switch can determine its forwarding path, and the remaining flits of the packet follow the same path. In this way, the CT switch doesn't need to wait until the whole packet is received, thus reducing the per-hop transmission delay significantly.

Although CT switching can significantly reduce transmission delay, it may not be easy to satisfy the low-latency requirements[2] without an accurate end-to-end latency prediction model. For example, admission control is imperative in guaranteeing quality-of-service (QoS) by allowing as many packet flows as the network can handle to satisfy any delay requirements, for which accurate end-to-end latency estimation is necessary. That is, with a proper end-to-end latency prediction method, a network operator can foresee the end-to-end latency performance after accepting a new flow. Unfortunately, however, no such method has been developed for CT switched networks with a random topology, since existing works on CT switching have considered deterministically-structured networks usually with homogeneous links [10], [11].

To fill the gap, this paper proposes an end-to-end latency prediction model for general-topology CT switching (GCS) networks. We first investigate how the end-to-end latency in the GCS network can be broken down into per-hop transmission and queueing delays. Based on the analysis, we present the method to predict each delay component of the end-to-end latency. In particular, we develop a method of predicting queueing delay by approximating the queueing model of GCS to the M/G/c queueing model.

The rest of the paper is organized as follows. Section II overviews related work, and Section III introduces the system model considered in this paper. Section IV proposes our end-to-end latency prediction model, and then Section V presents the prediction accuracy of the proposed method via extensive simulations based on an existing network topology. Finally, Section VI concludes the paper.

[1] In this paper, we use the terms 'router' and 'switch' interchangeably.

[2] The requirements include the average end-to-end latency and the packet delivery ratio (the portion of packets delivered in time), as introduced in [9].

II. RELATED WORK

A. QOS PROTOCOLS FOR LOW LATENCY SERVICES

One of the popular approaches to achieve low latency is to design a QoS protocol. There have been various quality of service (QoS) protocols proposed for low latency services in computer networks [12]. IntServ (Integrated Services) [13] reserves per-flow network resources via the Resource reSer-Vation Protocol (RSVP) [14] to guarantee minimal QoS for each flow, resulting in low and inflexible network utilization. DiffServ (Differentiated Services) [15], on the other hand, classifies packets into different QoS classes according to which the service level of each packet is determined, leading to coarse QoS provisioning. Furthermore, network slicing enables a network operator to tailor its network to the needs of low latency applications [16].

The GCS network considered in this paper achieves low latency performance in distinctive ways from the aforementioned approaches. Specifically, the GCS network neither reserves network resources nor categorizes packets into QoS classes. Instead, it achieves low end-to-end latency via flitization of each packet to alleviate per-hop transmission delay. Moreover, the GCS network does not require any logical separation of network resources as is the case in network slicing.

B. CUT-THROUGH SWITCHING IN THE LITERATURE

CT switching has been used in data center networks (DCN) [17] and Network-on-Chip (NoC) [18]. Most DCN switches support CT switching for low latency data transfer, while CT switching in NoC is implemented in the form of wormhole switching, a variant of CT switching with some implementation difference. Topologies of DCN and NoC, however, are homogeneous and well-structured in general, and hence existing analytic methods to predict end-to-end latency in DCN and NoC cannot be directly applied to the GCS network.

There have been just few work on CT switching in general-topology networks. Myrinet [19] is a high speed gigabit LAN built upon CT switching, and Autonet [20] adopts CT switching to accomplish per-hop packet forwarding within 2 ms. In addition, Time-Triggered Ethernet [21] routes all time-triggered messages via CT switching. As described, CT switching has been usually adopted for time-sensitive networks with stringent latency requirements.

Unfortunately, however, there exists no analysis on end-to-end latency prediction in heterogeneous CT switching network. Among the work related to end-to-end latency analysis, quasi cut-through switching (QCTS) [10] assumed over-simplified network conditions like uniform link capacities, while general cut-through switching (GCTS) [11] assumed matched link capacities between directly connected nodes. On the contrary, this paper assumes a more generalized model with heterogeneous links and random topology.

C. NETWORK CALCULUS

Network calculus [22] is one of the approaches that can be used for the network delay analysis. For instance, it has been used to model QoS parameters in wireless networks [23], [24]. Although network calculus can analyze the end-to-end latency based on a more realistic system model than queueing theory can [25], it can only estimate the latency bound (i.e., not the latency itself) thus limiting its usage.[3] To avoid such problems, this paper adopts queueing theory as a method to estimate the end-to-end latency.

III. SYSTEM MODEL

This section introduces the baseline architecture of the GCS network considered in this paper.

A. THE GCS NETWORK ARCHITECTURE

This paper considers a GCS network consisting of CT switching switches to serve latency-sensitive services. Each packet is assumed to be flitized at its source router where it is initially generated, forwarded through intermediate routers in the form of flits, and re-assembled into the original packet at the destination router. The considered model can be encountered in many real scenarios, e.g., DCN with heterogeneous links, a CT switching domain within the 5G network core, etc. We also suppose that link capacity is roughly proportional to the traffic intensity going through each link so that links at the network core have larger capacity than those at the network edges, which is reasonable when a network service provider (NSP) provisions its network according to per-link utilization. In addition, according to the link capacity suggestion, we assume that each edge link rates are similar to each other with small variance.

We assume each router in the GCS network consists of n input links, n input queues, n output links, n output queues, and a crossbar switch with speed-up of n. Assuming sufficiently large queue sizes, queuing overflow is not considered.[4] In addition, the switching latency in the crossbar switch (which is in the order of hundreds of nanoseconds [26]) is ignored because it is usually much smaller than the end-to-end latency requirements of low latency services [1]–[6].

This paper assumes Poisson packet arrivals at the source router while assuming a general (i.e., arbitrary) packet length distribution. In addition, we assume the packet length distribution is independent of the packet arrival process. Since this paper is the first attempt to built an analytical framework for end-to-end latency prediction in the GCS network, we believe the considered traffic model will become the cornerstone of the future research activities while still providing useful intuitions. In such a view, generalizing the packet arrival process is left as our future work.

[3]For example, when flow admission control is performed based on the upperbound of latency, many flows can be rejected unnecessarily leading to network under-utilization.

[4]Note that we will empirically show that this assumption is reasonable in Section V-A.

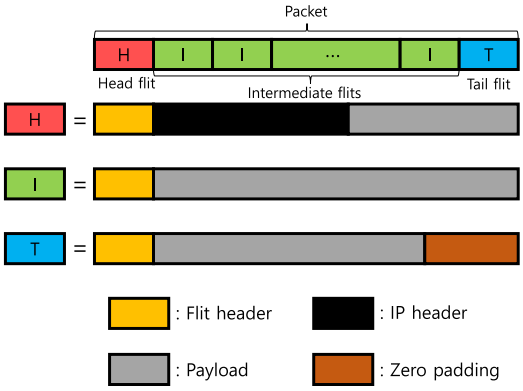


FIGURE 2. Flitization of a packet.

We also assume the NSP of the GCS network has a method to collect traffic information from its network, e.g., through software-defined networking (SDN) protocols [27]. For example, the SDN controller can collect information from each CT switching including packet arrival rates, packet length distribution, and per-flow routing path, which can be further distributed to selected switches so that they can utilize them for their flit scheduling and forwarding. In [28], it has been shown that using SDN helps to monitor and supervise network traffic easily.

B. CUT-THROUGH SWITCHING METHODOLOGIES

In the GCS network, each packet is split into same-sized flits at its source router, where the flit size is denoted by S_f (bits). The flits are classified into three categories: head flit, intermediate flit, and tail flit, as shown in Fig. 2. Among them, the head flit includes an IP packet header so that the switch can determine the routing path upon the arrival of a head flit, and the tail flit is zero-padded to have the same flit size as others.

Every flit has a 3 byte long flit header which includes the flit preamble, the packet identifier, and the flit's sequence number. The flit preamble (0.5 bytes) is used to identify the existence of a flit. The packet identifier (2 bytes) indicates which packet the related group of flits are belonging to. The flit sequence number (0.5 bytes) implies the ordered index of the flits from the same packet. Note that S_f should be larger than 63 bytes because the head flit must include an IP header, whose maximum size is 60 bytes with the options field.

The GCS network is assumed to be operated on a synchronized slotted time structure, where the time slot size S_s is determined as

$$S_s = S_f / r^* \text{ (sec)}, \tag{1}$$

where r^* (in bps) is the least common multiple of all the link rates in the GCS network. Note that time synchronization among switches can be done by existing protocols like the Precision Time Protocol (PTP) [29].

C. FLIT SCHEDULING MECHANISM

In this paper, a session is defined as a logical connection established between two edge routers (a source router and a destination router) for packet flow. Note that in the SDN architecture, forwarding decisions are flow-based, instead of destination-based, and the decisions are made by the SDN controller [27]. Therefore, we assume that the SDN controller determines the routing path of a session, and that the packets belonging to the same session follow the same path.

We consider Per-Session Bottleneck link Rate (PSBR) based scheduling for CT packets. The PSBR of a session i, denoted by γ_i, is the minimum link rate among the links on the session i's route such as

$$\gamma_i = \min_{l \in \mathcal{L}_i} r_l \text{ (flits/slot)}, \tag{2}$$

where i is the session index, $\mathcal{L}_i$ is the set of link indices on session i's route, and r_l is link l's rate. We assume that the SDN controller notifies the switches involved with a session of the session's PSBR in advance. Therefore, a switch can identify the PSBR of a packet by obtaining the session-related information (e.g., source and destination addresses) when the packet's head flit arrives.

The psuedo-code of the PSBR-based scheduling algorithm is described in Algorithm 1. At each packet arrival/departure, the queue checks whether a new packet can be served (lines 12–20 and 29–38). After a completion of packet transmission (line 27), if a queued packet can be served (line 33), the packet's counter is set to the maximum counter value plus 1 to keep the periodicity of ongoing flit transmissions (line 34). On the other hand, if a newly arriving packet can be served immediately (line 17), it means that the server of the queue is not fully utilized yet and thus the packet's counter is set to 0 (line 18) to avoid a possible unnecessary scheduling delay. Each counter decrements by 1 at each time slot (line 41). The server transmits the flit of the packet with the minimum counter value, which must be zero or less (lines 23–24), while updating its counter by adding the reciprocal of its PSBR (line 26). Note that a collision between the packets with the same minimum counter value is resolved by FIFO at line 23, where *arg min* gives the set of packets with the minimum counters and min returns the oldest packet among them. Such counter-based scheduling ensures each packet can be transmitted at its PSBR.

To better understand PSBR-based flit scheduling, let us consider an example shown in Fig. 3, where there exist two packets in an output queue, each belonging to one of the two distinct sessions with PSBR of 1/3 and 1/4, respectively. Assume the outgoing link rate is 1 (flits/slot), and the head flit of packet 1 arrives at time $t = 0$ while that of packet 2 arrives at time $t = 1$. With PSBR-based scheduling, the router sends the first flit of packet 1 at $t = 1$ and the second flit of packet 1 after 3 slots later, to follow the PSBR of 1/3. Similarly, the router sends the first flit of packet 2 at $t = 2$ and the second flit of packet 2 after 4 slots.

Algorithm 1 PSBR-Based Scheduling at an Output Queue

```
1  t ← 0;                                              // time slot index
2  K ← ∅;                                              // set of queued packets
3  S ← ∅;                                              // set of serving packets
4  k ← 0;                                              // number of accumulated packet arrivals
5  while t ≥ 0 do
6      if n packets arrive then
7          K ← K ∪ {k + 1, ..., k + n};
8          for w ← k + 1 to k + n by 1 do
9              b_w = FindPSBR(w) ;                     // PSBR of packet w
10         end
11         k ← k + n;
12         while K ≠ ∅ do
13             k_m ← min_{k'∈K} k';                    // oldest queued packet
14             if Σ_{s∈S} b_s + b_{k_m} > r_l then
15                 break;
16             end
               /* start to serve packet k_m                                */
17             S ← S ∪ {k_m};
18             c_{k_m} ← 0;                            // set the scheduling counter
19             K ← K \ {k_m};
20         end
21     end
       /* find the oldest packet among those with the min.
          counter                                                          */
22     c_min ← min_{s∈S} c_s;
23     s* ← min{arg min_{s∈S} c_s};
24     if c_min ≤ 0 and the slot is idle then
25         start to transmit the oldest flit of packet s*;
26         c_{s*} ← c_{s*} + 1/b_{s*};                 // update the counter
27         if the flit is the tail flit then
28             S ← S \ {s*};
               /* serve a new packet among queued                          */
29             while K ≠ ∅ do
30                 k_m ← min_{k'∈K} k';
31                 if Σ_{s∈S} b_s + b_{k_m} > r_l then
32                     break;
33                 else
34                     c_{k_m} ← max_{s∈S} c_s + 1;
35                     S ← S ∪ {k_m};
36                     K ← K \ {k_m};
37                 end
38             end
39         end
40     end
41     c_s ← c_s − 1,   ∀s ∈ S;                        // decrement all the counters
42     t ← t + 1;                                      // go to the next slot
43 end
```

The switch running the algorithm needs to know the PSBR of every session going through it, for which the SDN controller can collect all the link rates in the network and distribute them to the switches being managed. Hence, how often

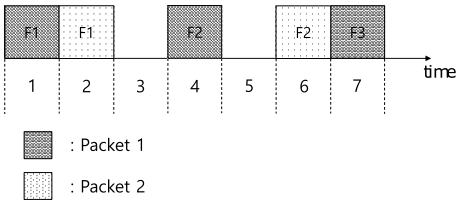


FIGURE 3. An example of PSBR-based flit scheduling.

such collection and distribution may incur determines the practicality of running the algorithm. Fortunately, link rates change only occasionally due to the installation/replacement of links, link failures, etc., thus preserving the practicality.

The motivation behind choosing the PSBR-based scheduling scheme is as follows. First of all, our scheme is flow-based scheduling, which is popular for scheduling in DCN and NoC [30], [31]. Unlike in DCN and NoC, however, it is inevitable for a GCS network to have a bottleneck link on an end-to-end routing path due to heterogeneous link rates. In such a case, the flits arriving at the bottleneck link faster than it can handle should lead to excessive queuing while unnecessarily consuming the network resources in the upstream routers. Hence, it is reasonable to force every router in a routing path to forward flits at the PSBR so that the remaining resources can be shared by other sessions. Note that such a bottleneck-rate driven forwarding strategy is commonly found in classical congestion control, e.g., leaky bucket traffic shaping, so as to reduce end-to-end latency.

To further clarify the aforementioned motivation, an example is provided in Fig. 4. Let us consider the network shown in Fig. 4(a), where there exist two packets at R_1 destined to R_3 and R_4 respectively as illustrated in Fig. 4(b). Under the FIFO scheduling, the blue packet is blocked until the red packet is completely transmitted from R_1 as seen from Fig. 4(c), which is not desirable since l_2 is much slower than l_1. That is, the 'I' flit of the red packet can be sent at slot 5 (not at slot 2) from R_1 with no penalty, since the transmission of the red 'H' flit at R_2 takes four slots. Hence, our scheme in Fig. 4(d) transmits each flit by its PSBR so that the blue packet's flits can be transmitted much earlier than in Fig. 4(c) resulting in smaller end-to-end latency (8 slots vs. 10 slots).

We should admit that there can be better scheduling schemes for a GCS network than the PSBR-based scheme. This paper, however, tries to provide a novel and generic methodology to analyze the end-to-end latency in CT switching networks, which would become a cornerstone of the further development in this field. Nevertheless, the proposed model may need to be fine-tuned to a given (non-PSBR-based) scheduling scheme.

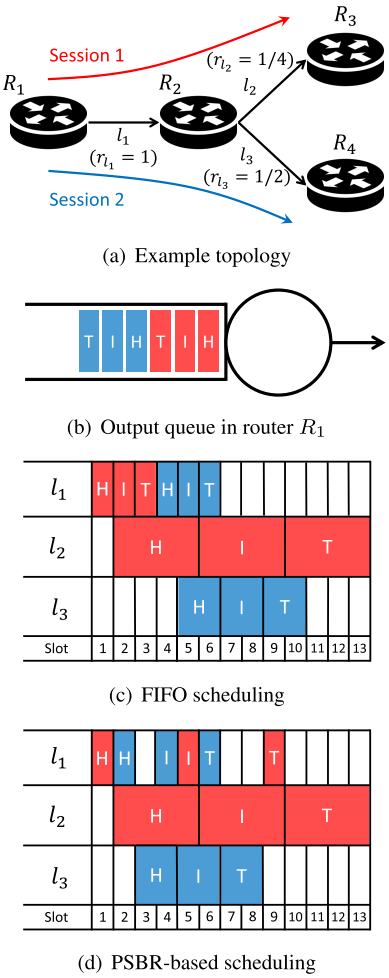


FIGURE 4. A motivating example of adopting the PSBR-based scheduling.

TABLE 1. A summary of notations.

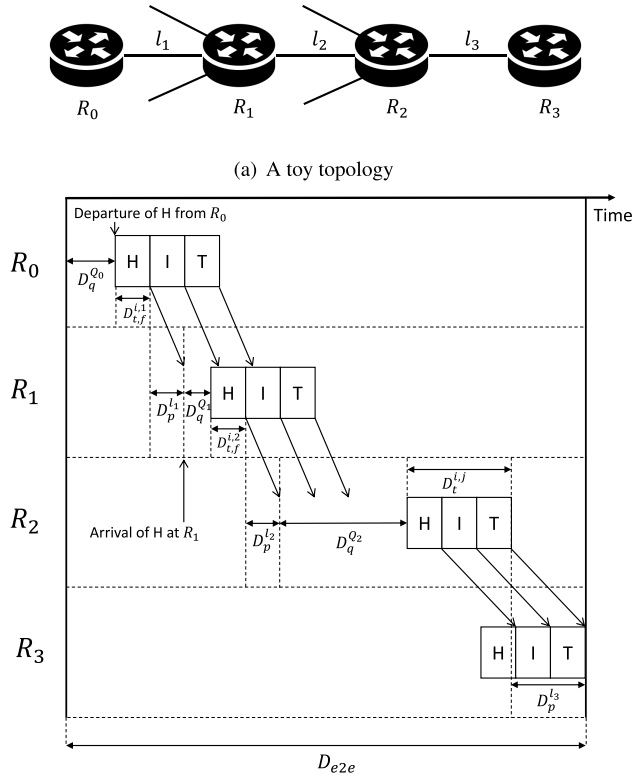
Notation	Definition
$D_{e2e}^{i,j}$	The end-to-end latency of packet j of session i
$D_q^{Q_k}$	The queueing delay of the head flit at the output queue of router k
$D_{t,f}^{i,k}$	The transmission delay of the head flit from router $k-1$ to k
$D_t^{i,j}$	The delay in transmitting all the flits of packet j at the last hop including per-flit transmission delay and the inter-flit wait-time
$D_p^{l_k}$	The propagation delay through the link l_k connecting routers $k-1$ and k
L_{l_k}	The physical length of link l_k
c_{l_k}	The travelling speed of a signal through l_k
r_{l_k}	Link l_k's rate
γ_i	PSBR of session i
$n_f^{i,j}$	The number of flits of packet j belonging to session i

IV. END-TO-END LATENCY PREDICTION METHOD

In this section, we propose a prediction method for the end-to-end latency in the GCS network. For better readability, the major notations introduced in this section are summarized in Table 1.

A. END-TO-END LATENCY BREAKDOWN ANALYSIS

We first investigate the factors contributing to the end-to-end latency. We define the end-to-end latency as the time interval between t_s and t_d, where t_s is the time when a head flit starts to be queued at the source router's output queue



(a) A toy topology

(b) end-to-end latency breakdown

FIGURE 5. An example of the end-to-end latency breakdown analysis.

and t_d is the time when a tail flit arrives at the destination router. It is assumed that the processing delay in flitization is negligible compared to transmission and queueing delays, as is generally true in modern routers. Therefore, we try to break down the end-to-end latency into queueing delays, transmission delays, and propagation delays.

To develop a basic understanding of the end-to-end latency in the GCS network, suppose a simple example of a three-hop network as depicted in Fig. 5(a). Then, Fig. 5(b) shows an end-to-end flit forwarding procedure of a packet in the considered network, where H, I, and T denote the head flit, an intermediate flit, and the tail flit of the packet, respectively. Note that we assume PSBR is one in this example.

We assume the head flit is initially output-queued at router R_0 during $D_q^{Q_0}$ slots (due to multiplexing with other packet flows, etc.). After that, it is transmitted during $D_{t,f}^{i,1}$ slots and traverses link l_1 consuming $D_p^{l_1}$ slots. When the head flit arrives at router R_1, it is queued for $D_q^{Q_1}$ slots, and subsequently the head flit is transmitted during $D_{t,f}^{i,2}$ slots. After spending $D_p^{l_2}$ slots for propagation, it arrives at router R_2, which is the last router prior to the destination. After the queueing of the head flit for $D_q^{Q_2}$ slots, all the flits are transmitted during $D_t^{i,j}$ slots and traverse link l_3 consuming $D_p^{l_3}$ slots. Based on this observation, the end-to-end latency of packet j of session i in the GCS network, denoted by $D_{e2e}^{i,j}$, can be generalized as

$$D_{e2e}^{i,j} = \sum_{k=1}^{n-1} D_{t,f}^{i,k} + D_t^{i,j} + \sum_{k=0}^{n-1} D_q^{Q_k} + \sum_{k=1}^{n} D_p^{l_k}, \tag{3}$$

where routers $0, 1, \ldots, n$ are on the routing path of the session i in the order of the indices, l_k is the link index, $D_q^{Q_k}$ is the queueing delay of the head flit at the output queue of router k, $D_{t,f}^{i,k}$ is the transmission delay of the head flit from router $k-1$ to k, $D_t^{i,j}$ is the delay in transmitting all the flits of packet j at the last hop including per-flit transmission delay and the inter-flit wait-time to satisfy the given PSBR, and $D_p^{l_k}$ is the propagation delay through the link l_k connecting routers $k-1$ and k. Note that Eq. (3) still holds for PSBR $\neq 1$, since PSBR only affects $D_t^{i,j}$.

Each term in Eq. (3) can be further derived as follows. First, $D_p^{l_k}$ (slots) is the time interval between when the first bit of the head flit leaves router $k-1$ and when it arrives at router k, and thus it is determined as

$$D_p^{l_k} = L_{l_k}/c_{l_k}, \tag{4}$$

where L_{l_k} (meters) is the physical length of link l_k and c_{l_k} (meters/slot) is the travelling speed of a signal through l_k. Note that $D_p^{l_k}$ is *deterministic* since L_{l_k} and c_{l_k} do not vary. Next, $D_{t,f}^{i,k}$ (slots) is the time required to transmit all the bits of the head flit, thus obtained by

$$D_{t,f}^{i,k} = \frac{1}{r_{l_k}}, \tag{5}$$

which is also *deterministic*. In addition, $D_t^{i,j}$ (slots) is the entire time to transmit all the flits of a packet at the last hop including the pausing intervals due to PSBR, which is given as

$$D_t^{i,j} = \frac{n_f^{i,j} - 1}{\gamma_i} + \frac{1}{r_{l_n}}, \tag{6}$$

where $n_f^{i,j}$ is the number of flits of packet j. Given session i, $D_t^{i,j}$ is *random* since $n_f^{i,j}$ is randomly varying with packets.

Lastly, $D_q^{Q_k}$ is the time interval between when the head flit is firstly queued at queue Q_k and when the head flit starts to be transmitted, which is *stochastic* by nature. Conventional approaches to estimate queueing delays, however, cannot be directly applied to our problem. Suppose a flit is a schedulable job of the queueing model. Under the PSBR-based scheduling, job inter-arrival times are highly dependent on each other for the jobs (i.e., flits) belonging to the same packet. That is, we need a queueing model that can describe a hierarchical structure between packets and flits, which has not been developed so far. In the following section, we propose a method to resolve this issue.

Finally, the expected end-to-end latency $E[D_{e2e}^{i,j}]$ can be derived by replacing $n_f^{i,j}$ in Eq. (6) with $E[n_f^{i,j}]$ and $D_q^{Q_k}$ in Eq. (3) with with $E[D_q^{Q_k}]$. $E[n_f^{i,j}]$ is easily obtained from the packet length distribution per packet, while $E[D_q^{Q_k}]$ is still

unknown. The following section IV-B will discuss how we can approximate $E[D_q^{Q_k}]$ by tailoring a traditional queuing model to our problem.

B. M/G/C-BASED APPROXIMATE QUEUEING MODEL

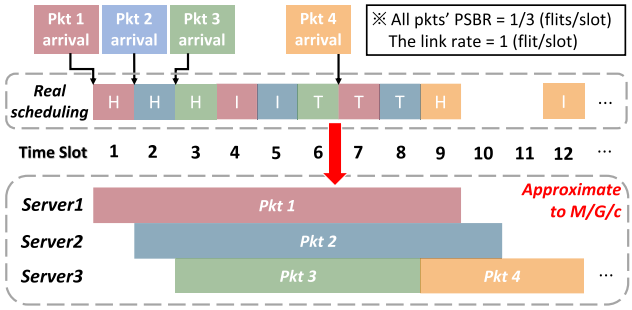
As discussed in Section IV-A, we need a new queueing model for the head flit's per-hop queueing delay. To do so, we treat the head flit as a schedulable job whose service time is the whole packet service time transmitting from the head flit to the tail flit of the packet they belong to. That is, all the flits of a packet are considered as one logical job at a given queue, although they may be in fact transmitted to the outgoing link at different times intermittently. More formally, the service time for packet j of session i is defined as $n_{f,j}/\gamma_i$ (slots) where $n_{f,j}$ is the number of flits of packet j. Accordingly, a job arrival occurs when the head flit of a new packet arrives.

Then, assuming Poisson packet arrivals at source routers as mentioned earlier in Section III, we further conjecture that the head-flit arrival at intermediate routers follows the Poisson process as well. The conjecture is based on the assumption that the packet arrivals at intermediate routers may be independent of the packet length distribution, if the Kleinrock independence approximation assumption [32] can still be applied to the GCS network. Although there is a lack of relevant studies to confirm the conjecture, later on in Section V we will provide some experimental evidence to support it.

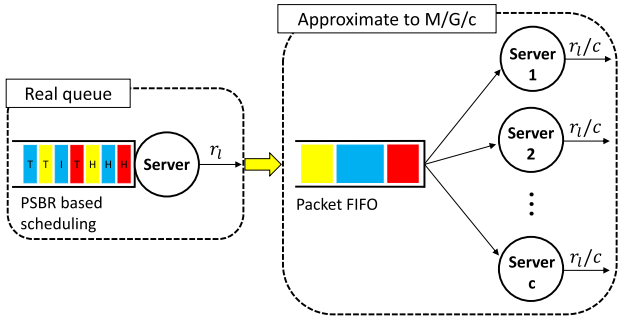
In addition, under the PSBR-based scheduling, each queue can be approximated to a multi-server queue. As assumed in Section III, the link rate is smaller at network edges than at the network core, and the edge link rates are roughly similar to each other with small variance. Then, most sessions would have their bottleneck link rates as their edge link rates, and thus their PSBRs become similar. As a result, when the sessions multiplexed at an output queue have similar PSBRs, the queue approximately acts like a multi-server queue with c servers, each with identical service rate of $\mathcal{R}_Q$ (flits/slot), the average value of PSBR of all sessions passing through the queue, where c is roughly given as the output link rate divided by $\mathcal{R}_Q$. Note that c will be more rigorously determined in the sequel.

Based on the aforementioned discussions, we approximate an output queue in the GCS network to a M/G/c queueing model. Then, we adopt the M/G/c queueing delay approximation scheme in [33] to derive the head flit's approximate queuing delay. Note that since the original M/G/c queueing model is a continuous-time model while our system operates in discrete-time with time slots, we adjust the M/G/c queueing model accordingly in the sequel to cope with the slotted nature. Fig. 6(a) depicts the logical concept of the new queueing model under the PSBR-based scheduling scheme, and Fig. 6(b) illustrates how a real queue is approximated to an M/G/c queue.

We start with defining some notations for a given output queue Q_k. Let $p_q^{Q_k}(x)$ denote the probability mass



(a) Real scheduling vs. approximate M/G/c based scheduling



(b) The schematic figure of a real queue vs. an approximate M/G/c queue

FIGURE 6. The concept of the proposed queueing model.

function (PMF)[5] of the queueing delay $D_q^{Q_k}$, $V^{Q_k}(x)$ denote the waiting time distribution of delayed packets such as

$$V^{Q_k}(x) = P(D_q^{Q_k} \leq x | D_q^{Q_k} > 0), \tag{7}$$

and $p_w^{Q_k}$ denote the delay probability defined as

$$p_w^{Q_k} = P(D_q^{Q_k} > 0). \tag{8}$$

Then, the cumulative distribution function (CDF) of $D_q^{Q_k}$, denoted by $F_q^{Q_k}(x)$, can be expressed as

$$\begin{aligned} F_q^{Q_k}(x) &= P\left(D_q^{Q_k} \leq x\right) \\ &= P\left(D_q^{Q_k} = 0\right) + P\left(0 < D_q^{Q_k} \leq x\right) \\ &= (1 - p_w^{Q_k}) + V^{Q_k}(x) \cdot p_w^{Q_k}. \end{aligned} \tag{9}$$

In addition, we denote by λ_{Q_k} the head flit arrival rate (in flits/slot) and by S_{Q_k} the (random variable of) service time (in slots) of each job in our queueing model. Moreover, c_{Q_k} is the approximate number of servers, which is determined as

$$c_{Q_k} = \left\lfloor \frac{r_{Q_k}}{\mathcal{R}_{Q_k}} + \frac{1}{2} \right\rfloor, \tag{10}$$

where r_{Q_k} is the outgoing link rate, and $1/2$ is to round up the value inside the floor operation (called *round half up*). Then, the traffic intensity ρ_{Q_k} at queue Q_k, which is defined as the ratio of time a server is occupied *to* the time the server is available [34], is obtained by

$$\rho_{Q_k} = \frac{\lambda_{Q_k} E[S_{Q_k}]}{c_{Q_k}}, \tag{11}$$

[5]We consider PMF instead of probability density function since time is discretized into slots.

following the traffic intensity formula of the conventional multi server queueing model [34].

Now, according to [33], $V^{Q_k}(x)$ can be approximated to $\overline{V}^{Q_k}(x)$ such as

$$\begin{aligned}&\overline{V}^{Q_k}(x)\\&= (1-\rho_{Q_k})\{1-(1-F_{eq}^{Q_k}(x))^{c_{Q_k}}\}\\&\quad+\lambda_{Q_k}\int_0^x \overline{V}^{Q_k}(x')\{1-F_{S_{Q_k}}\left(c_{Q_k}(x-x')\right)\}dx',\end{aligned}\tag{12}$$

where $F_{eq}^{Q_k}$ is defined as

$$F_{eq}^{Q_k}(x)=\frac{1}{E[S_{Q_k}]}\sum_{t=0}^{x}\left(1-F_{S_{Q_k}}(t)\right)\tag{13}$$

and $F_{S_{Q_k}}$ is the CDF of S_{Q_k}. Note that the summation in Eq. (13) is to consider the nature of slotted time. Note that Eq. (12) can be evaluated by numerical methods like the rectangle method [35]. In addition, $p_w^{Q_k}$ can be approximated to $\overline{p}_w^{Q_k}$ [33] such as

$$\overline{p}_w^{Q_k}=\frac{\left(\lambda_{Q_k}E[S_{Q_k}]\right)^{c_{Q_k}}}{c_{Q_k}!(1-\rho_{Q_k})}\overline{p}_0,\tag{14}$$

where

$$\overline{p}_0=\left(\sum_{j=0}^{c_{Q_k}-1}\frac{\left(\lambda_{Q_k}E[S_{Q_k}]\right)^j}{j!}+\frac{\left(\lambda_{Q_k}E[S_{Q_k}]\right)^{c_{Q_k}}}{c_{Q_k}!\left(1-\rho_{Q_k}\right)}\right)^{-1}.\tag{15}$$

Using Eqs. (12) and (14), $F_q^{Q_k}(x)$ can be approximated to $\overline{F}_q^{Q_k}(x)$ such as

$$\overline{F}_q^{Q_k}(x)=(1-\overline{p}_w^{Q_k})+\overline{V}^{Q_k}(x)\ \cdot\overline{p}_w^{Q_k}.\tag{16}$$

Finally, the end-to-end latency distribution can be estimated as follows. Let $p_e^i(x)$ be the PMF of $D_{e2e}^{i,j}$ for session i. Remind that the terms $D_p^{l_k}$ and D_t^i in $D_{e2e}^{i,j}$ are deterministic whereas the other two terms are random. Let $p_{n_f^{i,j}}(x)$ be the PMF of $n_f^{i,j}$. Then, the PMF of $D_t^{i,j}$, denoted by $p_t^i(x)$, can be derived as

$$\begin{aligned}p_t^i(x)&=P\left(D_t^{i,j}=x\right)=P\left(\frac{1}{\gamma_i}(n_f^{i,j}-1)+\frac{1}{r_{l_n}}=x\right)\\&=P\left(n_f^{i,j}=\gamma_i\left(x-\frac{1}{r_{l_n}}\right)+1\right)\\&=p_{n_f^{i,j}}\left(\gamma_i\left(x-\frac{1}{r_{l_n}}\right)+1\right).\end{aligned}\tag{17}$$

Assuming the independence among queues, we can obtain $p_e^i(x)$ such as

$$\begin{aligned}p_e^i(x)&=P\left(D_{e2e}^{i,j}=x\right)\\&=P\left(\sum_{k=1}^{n-1}D_{t,f}^{i,k}+D_t^{i,j}+\sum_{k=0}^{n-1}D_q^{Q_k}+\sum_{k=1}^{n}D_p^{l_k}=x\right)\\&=P\left(D_t^{i,j}+\sum_{k=0}^{n-1}D_q^{Q_k}=\dot{x}\right)\\&=p_q^{Q_0}(\dot{x})*p_q^{Q_1}(\dot{x})*\cdots*p_q^{Q_{n-1}}(\dot{x})*p_t^i(\dot{x})\\&\approx\overline{p}_q^{Q_0}(\dot{x})*\overline{p}_q^{Q_1}(\dot{x})*\cdots*\overline{p}_q^{Q_{n-1}}(\dot{x})*p_t^i(\dot{x})=\overline{p}_e^i(x)\end{aligned}\tag{18}$$

where $\dot{x}=x-(\sum_{k=1}^{n-1}D_{t,f}^{i,k}+\sum_{k=1}^{n}D_p^{l_k})$, $*$ denotes convolution, $\overline{p}_q^{Q_k}(x)$ is an approximation of $p_q^{Q_k}(x)$ which can be obtained from the approximate CDF in Eq. (16), and $\overline{p}_e^i(x)$ is an approximation of $p_e^i(x)$.

Based on the analysis, various end-to-end latency metrics can be estimated. For example, the average end-to-end latency of session i, i.e., $E[D_{e2e}^{i,j}]$, can be estimated as

$$E[D_{e2e}^{i,j}]=\sum_{x=1}^{\infty}xp_e^i(x)\approx\sum_{x=1}^{\infty}x\overline{p}_e^i(x).\tag{19}$$

In addition, the packet delivery ratio (PDR) of session i, defined as the ratio of packets satisfying their delay requirement D_{req}^i (which is the maximum allowed end-to-end latency of session i's packet), is obtained by

$$PDR_i=F_e^i(D_{req}^i)\approx\overline{F}_e^i(D_{req}^i)\tag{20}$$

where $F_e^i(x)$ is the CDF of the end-to-end latency of session i and $\overline{F}_e^i(x)$ is its approximation obtained from $\overline{p}_e^i(x)$.

C. ZERO-FORCING THE QUEUEING DELAY IN SPECIAL CASES

There exists a special case where the queuing delay in certain queues is always zero due to a topological issue in heterogeneous CT switching networks. To understand the phenomenon better, let us consider the tandem queue with two connected routers in Fig. 7, where the link rate of router R_2's output link l_2 is higher than or equal to that of router R_1's output link l_1. Considering the fact that the flit size is fixed, it is apparent that the transmission delay of each flit transmission at R_2 is always smaller than that at R_1. As a result, the flits transmitted by R_1 will never be queued at R_2 resulting in no queueing delay at R_2. Therefore, we need to manually set the queueing delay in such routers to zero. In Section V-C, we will show the efficacy of the proposed zero-forcing technique via simulations.

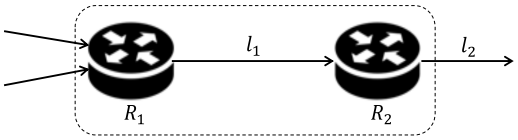


FIGURE 7. An example scenario for zero queueing delay.

The following theorem formally states the aforementioned case.

Theorem 1: In cut-through switching networks, when two tandem queues Q_1 and Q_2 are given where packet flows are moving from Q_1 to Q_2, the queueing delay at queue Q_2 becomes always zero if the link rate $r_{l_{out}}$ of its output link l_{out} is equal to or higher than $r_{l_{in}}$, the link rate of its input link l_{in}.

Proof: Suppose the kth flit arrives at Q_2 at t_0 (slot) when Q_2 is empty. Since the queue is empty, the flit is not queued. Then, the kth flit leaves Q_2 at $t_1 = S_f/r_{l_{out}} + t_0$, and the $(k+1)$th flit can arrive at Q_2 when $t_2 \geq S_f/r_{l_{in}} + t_0$. Since $S_f/r_{l_{in}} \geq S_f/r_{l_{out}}$, the $(k+1)$th flit will never be queued at Q_2. By the mathematical induction, no flit will be queued at Q_2. □

D. DISCUSSION

The end-to-end latency prediction for each session should be performed whenever there is any expected change in the stochastic behavior of delay components. As explained earlier, given session i, the stochastic nature of $D_t^{i,j}$ is invariant, while $D_p^{l_k}$ and $D_{t,f}^{i,k}$ are deterministic. Hence, $D_{e2e}^{i,j}$ related to Q_k needs to be re-estimated only if the stochastic properties of $D_q^{Q_k}$ is changed, which happens when any session passing through Q_k is added to or removed from the GCS network.

More specifically, suppose session η is established (or removed), router k is on the routing path of the session, and Q_k is the output queue in router k through which session η passes. Then, every other session i also passing through Q_k is affected by the change. Therefore, the SDN controller needs to re-estimate $D_{e2e}^{i,j}$ for each session i, for which the SDN controller needs the traffic information (e.g., packet arrival rate & packet length distribution) of session η and all sessions passing through Q_k. To do so, the traffic information of session η is reported to the SDN controller by the source router at the session's establishment, which is combined with previously acquired traffic information of all other sessions passing through Q_k to re-obtain $D_{e2e}^{i,j}$ for each i.

On the other hand, CT switching could be less reliable in error-prone networks since a packet cannot be recovered at the destination if one or more flits (of the packet) are received in error. In such a case, the source can retransmit only those flits in error or may retransmit the whole flits of the packet, depending on which retransmission strategy is used. Our proposed model can be slightly modified to address such a situation, by increasing packet arrival rate at the source as much as retransmission's contribution. Then, a packet's average end-to-end latency after retransmissions can be obtained by combining the average number of retransmissions with our model's predicted end-to-end latency of a single packet transmission (without retransmission). It should be noted, however, per-flit error probability is much smaller than per-packet error probability and thus the true impact of error on CT switching should be further investigated. Since the aforementioned issues deserve a separate research effort, we leave it as our future work.

V. PERFORMANCE EVALUATION

In this section, we evaluate the accuracy of the proposed end-to-end latency prediction model, where the accuracy is assessed by the error in predicting the latency, which is defined as

$$error := \frac{sim.value - pre.value}{sim.value} \cdot 100\ (\%) \qquad (21)$$

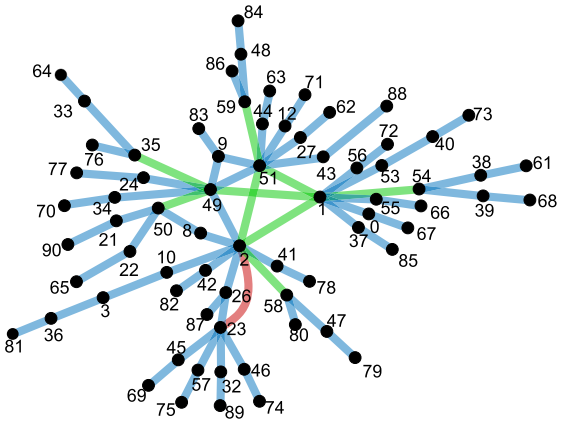


FIGURE 8. Modified RedCLARA topology.

where *sim.value* and *pre.value* denote the simulated latency and the predicted one by our model, respectively. We consider the average error and the 90th percentile error as performance measures, while simulating various flit sizes and traffic intensities. In addition, a simulation run lasts for 4 million slots, while the first 1 million slots are excluded from the accuracy assessment considering the time it takes to converge to the steady state. Finally, per-session routing path is obtained by running Dijkstra's algorithm, which is commonly applied to all simulation scenarios.

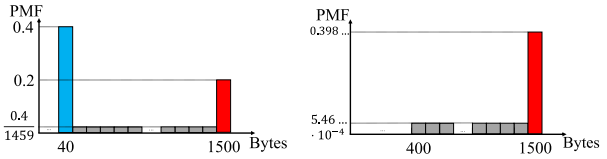
The simulated network topology is a modified version of RedCLARA, an academic data network established in Latin America [36]. Fig 8 illustrates the simulated topology which consists of 71 nodes (among which 30 nodes are at the network edge) and 76 links with varying capacity such as 100 Mbps (colored in blue), 200 Mbps (colored in green), or 400 Mbps (colored in red). Link lengths are assumed uniformly distributed for simplicity. Since each link is bi-directional and connected to two routers, there exist $152 (= 76 \cdot 2)$ output queues. We associate each of the output queues to (i) a queue index from 1 to 152, and (ii) a label in the form of OQ#A-B where A is the index of the router where the output queue resides and B is the index of the router the outgoing link from router A is connected to. In addition, each router at the network edge is considered as a source router of 20 sessions,[6] whose destination router is also chosen from the edge routers. Fig. 8 shows the network topology used in the simulation, with router indices specified.

In the simulations, packets follow the Poisson arrival process where the arrival rates of all the sessions are uniformly distributed as described in Table 2. In addition, we consider a modified Internet packet length distribution model as follows. According to [37], [38], the packet length distribution in the Internet presents a U-shaped pattern shown in Fig. 9(a). In GCS networks, however, small size packets like 40 Bytes won't be the target for flitization since concurrent transmission over multiple hops by CT switching is beneficial

[6]Note that assuming a larger number of sessions than 20 may give us better prediction results, due to the nature of the Kleinrock independence approximation.

TABLE 2. Simulation configurations: six scenarios.

Simulation index	# of sessions in the network	Packet arrival rate of each session i (pkts/sec)	Traffic intensity (ρ)	Flit size (Bytes)
Sim. 1	600	$\lambda_i \sim Unif[1000, 2000]$	Light ($\rho \in [0.2, 0.4]$)	100
Sim. 2	600	$\lambda_i \sim Unif[1000, 2000]$	Light ($\rho \in [0.2, 0.4]$)	200
Sim. 3	600	$\lambda_i \sim Unif[1000, 2000]$	Light ($\rho \in [0.2, 0.4]$)	300
Sim. 4	600	$\lambda_i \sim Unif[2000, 4000]$	Heavy ($\rho \in [0.4, 0.8]$)	100
Sim. 5	600	$\lambda_i \sim Unif[2000, 4000]$	Heavy ($\rho \in [0.4, 0.8]$)	200
Sim. 6	600	$\lambda_i \sim Unif[2000, 4000]$	Heavy ($\rho \in [0.4, 0.8]$)	300



(a) The Internet packet length distribution (b) A modified Internet packet length distribution

FIGURE 9. The packet length distribution used in simulations.

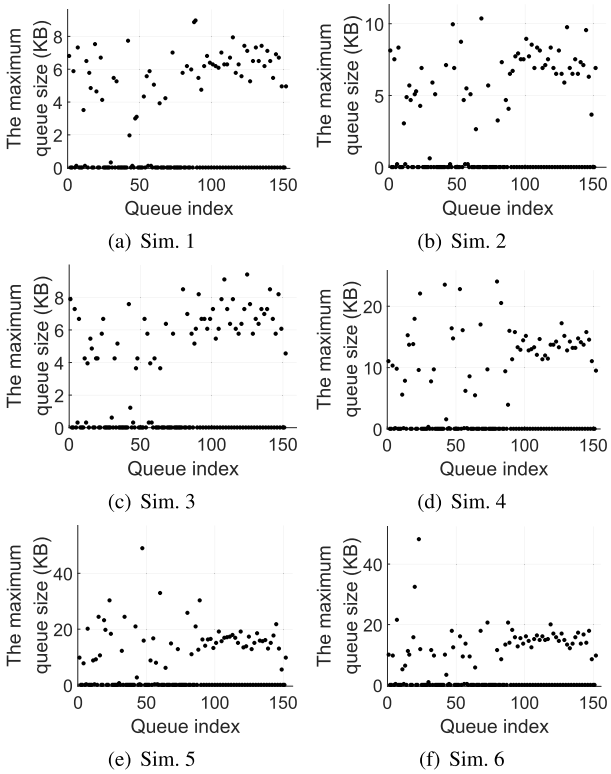


FIGURE 10. Per-queue maximum queue size observed.

for long packets as analyzed in [9]. Therefore, we modify the original distribution in Fig. 9(a) to the one shown in Fig. 9(b), by excluding the packets smaller than 400 Bytes and re-normalizing the distribution.

We tested six types of simulations with varying network traffic intensity and flit sizes. The flit size configuration affects the performance of the network, since too small flits may cause more flitization overhead (due to flit header at each flit) whereas too large flits may result in excessive zero padding at the tail flit. To capture the effect, we varied the flit size S_f as 100, 200, and 300 bytes. In addition, two traffic intensities are considered for the simulations – light vs. heavy traffic. Table 2 summarizes the configuration of the simulations.

A. QUEUE SIZE VARIATIONS

We first present the simulated results of the maximum queue size at each queue in Fig. 10, to experimentally verify the validity of our assumption in Section III-A, i.e., no queueing overflow. As seen, the queue size is effectively upperbound by a certain threshold in the order of a few KB to dozens of KB. Considering that state-of-the-art routers have the buffer size in the order of hundreds of KB to tens of MB [39], [40], our assumption of no queueing overflow seems reasonable.

B. POISSON HEAD FLIT ARRIVALS

This section investigates whether the head flit arrivals at intermediate routers indeed follow the Poisson process, as we assumed in Section IV-B. To do so, we measure the distribution of head flit inter-arrival times in all the intermediate routers, leading to a total of 352 distributions examined considering six simulation scenarios. Among them, it turns out 308 cases (87.5%) very accurately follow the Poisson distribution and 41 cases (11.6%) show distributions fairly similar to the Poisson distribution, while only 3 cases present distributions different from Poisson. As a result, we experimentally confirm that head flit arrivals follow the Poisson distribution *not only* at source routers *but also* at intermediate routers.

As an example, we show the distribution of two selected output queues OQ#2-1 and OQ#49-51, each located at a core router under the simulation scenarios 2 and 5 (i.e., light and heavy traffic respectively, but moderate flit size for both). Fig. 11 presents the sample distributions of their head flit inter-arrival times compared to the exponential distribution,[7] where each exponential distribution is fitted to the corresponding sample distribution via the minimum mean squared error estimation. As shown, the head flit inter-arrival times at the chosen routers are distributed almost exactly as exponential.

[7]Exponentially distributed and independent inter-arrival times instantly mean Poisson arrivals.

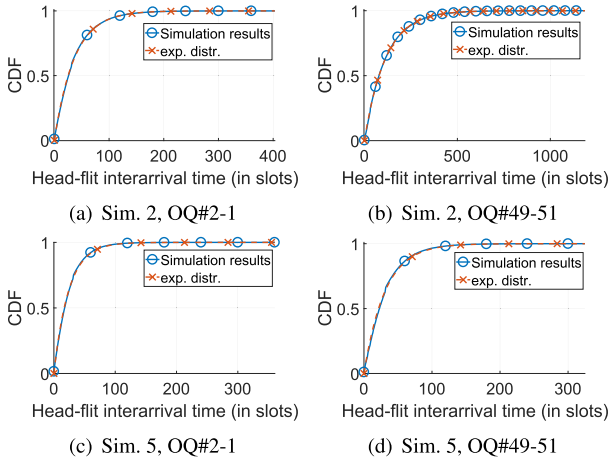


FIGURE 11. Head flit inter-arrival time distributions at OQ#2-1 and OQ#49-51.

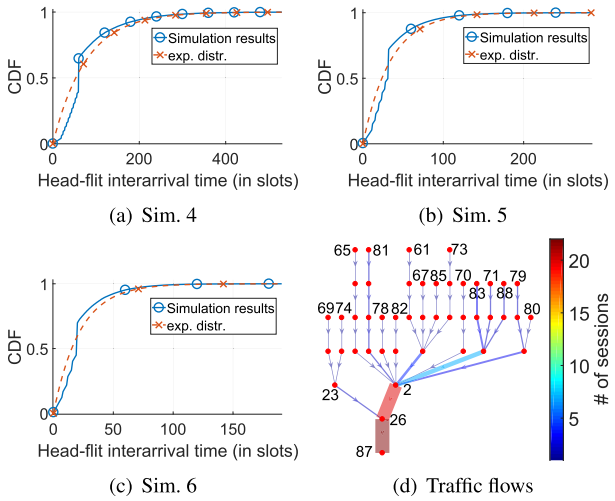


FIGURE 12. OQ#26-87: Head flit inter-arrival time distributions and its related traffic flows.

The aforementioned three non-exponential cases, in fact, result from the same output queue OQ#26-87 under the scenarios 4 to 6, which are shown in Fig. 12. The reason is that OQ#26-87 does not satisfy the necessary condition of the Kleinrock independence approximation assumption as explained in the following. Fig. 12(d) illustrates the track of sessions related to OQ#26-87, where each edge is a link, each node is a router, and both width and color of an edge implies the number of sessions going through the link. As can be seen, most traffics come from one dominant input link (the one between routers 2 and 26), and thus OQ#26-87 resembles linearly connected tandem queues.

C. QUEUEING DELAY PREDICTION

This section evaluates the accuracy of the M/G/c-based approximate queueing model and the validity of the zero-forcing technique. Although Section V-B has shown that the head flit arrivals follow the Poisson process at the intermediate routers, it does not guarantee the independence between packet arrivals and service times. Therefore, we need to investigate the prediction accuracy both at source routers and intermediate routers.

We start with investigating the queueing delay prediction results at the source routers. First, Fig. 13 compares the prediction result by the proposed method with the simulated result in terms of the queueing delay distribution at two source routers (OQ#89-32 and OQ#73-40) regarding Sim. 2 and 5. In addition, Fig. 14 illustrates the prediction error of the

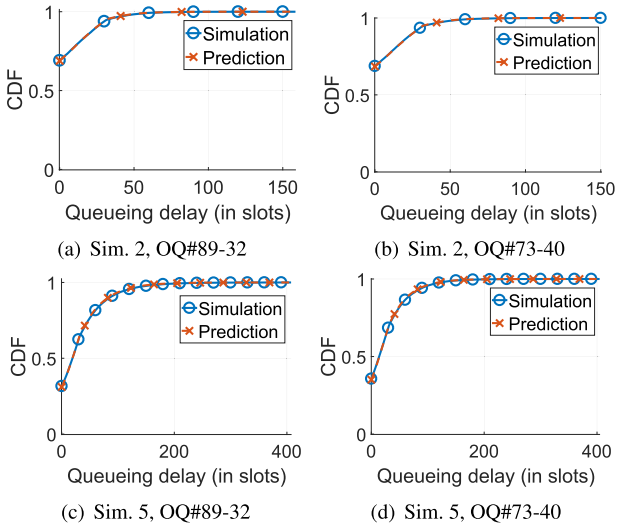


FIGURE 13. Queueing delay distribution at two source routers.

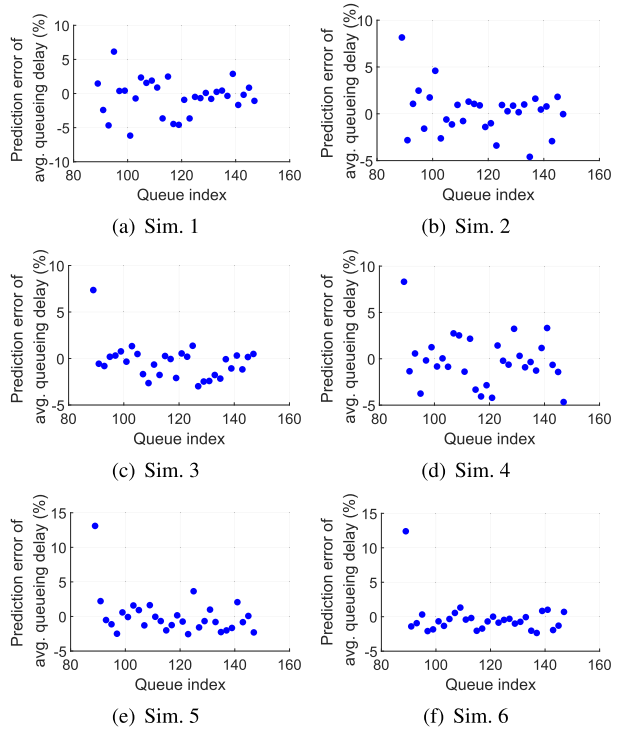


FIGURE 14. Prediction error in the average queueing delay at all source routers.

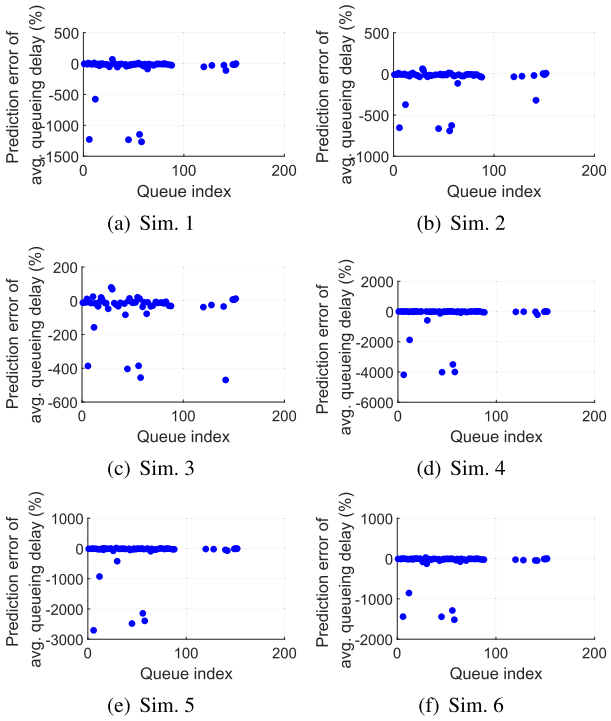


FIGURE 15. Prediction error in the average queueing delay at all intermediate routers.

average queueing delay at all 30 source routers for Sim. 1 through 6, respectively. As can be seen from both figures, the proposed method predicts the queueing delay very accurately. Specifically, we define a metric $A_{err}^{\mathcal{Q}}$ to evaluate the average performance such as

$$A_{err}^{\mathcal{Q}} = \frac{\sum_{q\in\mathcal{Q}} |PEA_q|}{|\mathcal{Q}|} \ (\%), \tag{22}$$

where $\mathcal{Q}$ is the set of given queues and PEA_q is the prediction error in the average queueing delay at queue q. Then, $A_{err}^{\mathcal{Q}}$ in Fig. 14 ranges from 1.287% to 2.004% (which is obtained by setting $\mathcal{Q}$ as the set of queues in the source routers).

Next, we also present the error in queueing delay prediction at all the intermediate routers, as shown in Fig. 15. While most of the prediction errors tend not too severe, there exist six outliers (queue indices 6, 12, 30, 45, 56, 58) with considerable errors in most simulation scenarios. The outliers have a common feature that the sum of input link rates is smaller than or equal to the output link rate to which the input links are destined. It should be noted, however, the contribution of such cases to the end-to-end latency is insignificant in our simulations since the queueing delay in those six queues are quite small compared to others. Additionally, the queue index 142, presents a relatively large error in Sim. 2 and 3., which is because the queue is in fact OQ#26-87 shown earlier in Fig. 12, the special case not following Poisson arrivals. Note that we leave it as our future work to further investigate how to alleviate the aforementioned outliers.

TABLE 3. $A_{err}^{\mathcal{Q}}$ for the intermediate routers excluding the six outliers.

Simulation	$A_{err}^{\mathcal{Q}}$ (%)
Sim. 1	18.8
Sim. 2	18.1
Sim. 3	24.1
Sim. 4	19.8
Sim. 5	17.5
Sim. 6	15.5

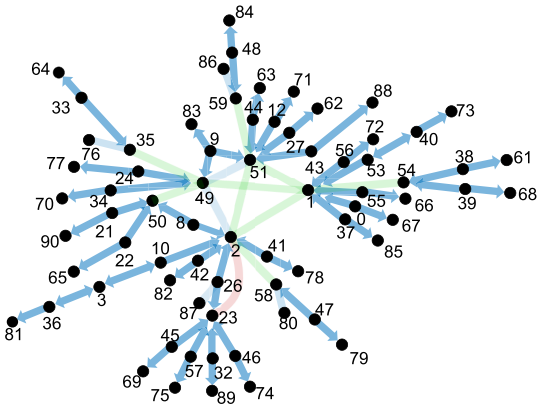


FIGURE 16. The queues to apply the zero-forcing technique shown in blue arrows.

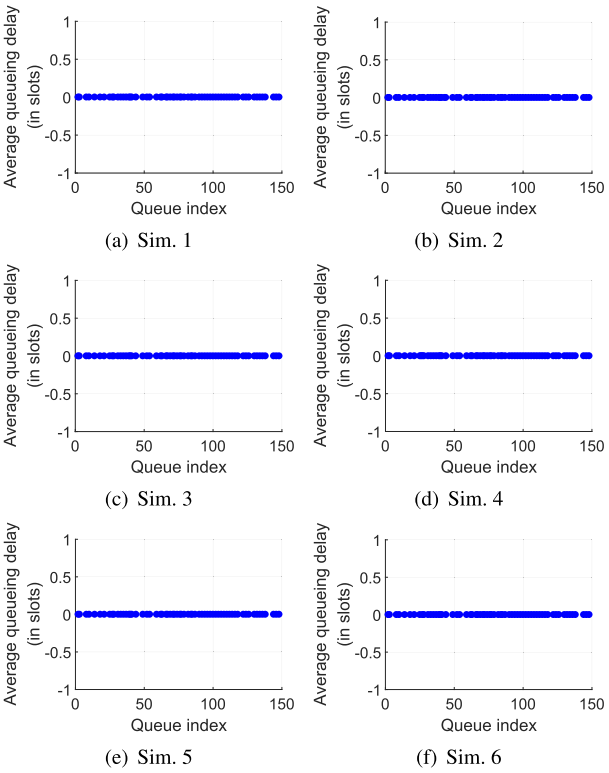
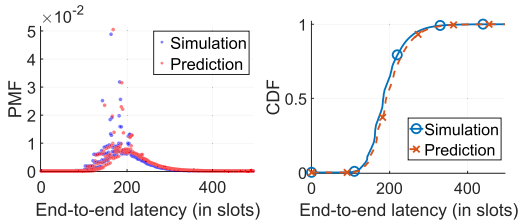
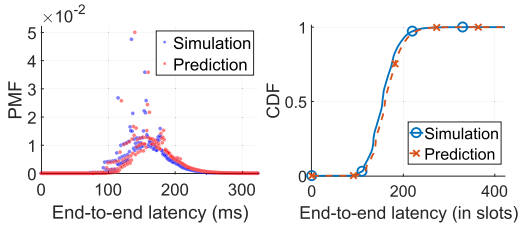


FIGURE 17. Average queueing delay at the queues with zero expected delay.

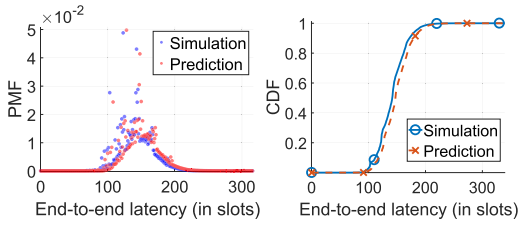
Excluding the six outliers with indices 6, 12, 30, 45, 56, 58, we measured $A_{err}^{\mathcal{Q}}$ of Fig. 15 as shown in Table 3, where $\mathcal{Q}$ is now the set of queues in the intermediate routers excluding



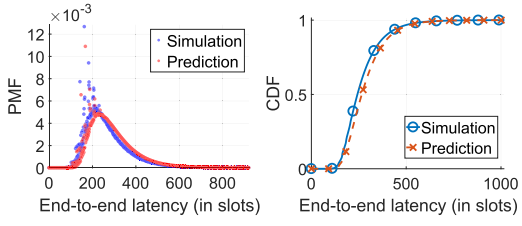
(a) Sim. 1, PMF(left) and CDF(right)



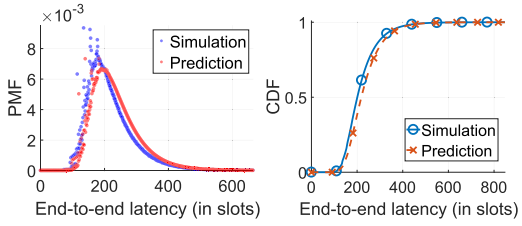
(b) Sim. 2, PMF(left) and CDF(right)



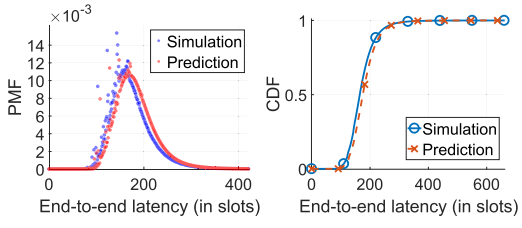
(c) Sim. 3, PMF(left) and CDF(right)



(d) Sim. 4, PMF(left) and CDF(right)



(e) Sim. 5, PMF(left) and CDF(right)



(f) Sim. 6, PMF(left) and CDF(right)

FIGURE 18. End-to-end latency distribution: proposed(labeled as 'Prediction') vs. simulated(labeled as 'Simulation').

the outliers. We can notice that the errors are larger than the ones at source routers, ranging from 15.5% to 24.1%. Nevertheless, we will later see in Section V-D that the proposed model still produces reasonably accurate results when predicting the end-to-end latency, due to the following reasons. As we analyzed in Section IV-A, the queueing delay is just one of various delay components contributing to the end-to-end latency. Indeed, we discovered from our simulations that the intermediate routers with less accurate prediction only partially contribute to the accumulated queueing delay. Therefore, the queueing delay prediction errors at intermediate routers are tolerable when it comes to predicting the end-to-end latency.

TABLE 4. Latency prediction accuracy by the proposed method.

Simulation	Error in $E[D_{e2e}^{i,j}]$ (%)	90th percentile error (%)
Sim. 1	-4.75	-5.26
Sim. 2	-4.43	-4.10
Sim. 3	-4.61	-4.65
Sim. 4	-7.65	-6.05
Sim. 5	-7.31	-5.77
Sim. 6	-5.47	-3.98

Regarding the proposed zero-forcing technique, Fig. 16 depicts the queues with zero expected queueing delay as a blue arrow (an arrow from node A to B implies OQ#A-B), while Fig. 17 presents the measured average queueing delay at such queues (63 out of 152 queues). As clearly seen, all the results present zero delay as expected, which shows that the proposed zero forcing scheme is correct and appropriate.

D. END-TO-END LATENCY PREDICTION

Using the proposed end-to-end latency prediction model, we predict the end-to-end latency distribution of all packets and compare it with the empirical distribution obtained by simulations. Fig. 18 presents the results for simulations 1 through 6. As summarized in Table. 4, the error in the average end-to-end latency prediction ranges from -4.43% to -7.65%. In addition, the 90th percentile errors are distributed from -3.98% to -6.05%. Overall, the proposed method turns out to be quite accurate in latency prediction, showing its efficacy as a basic building block in designing and managing future ultra-low latency applications in the GCS network.

VI. CONCLUSION

In this paper, we have considered packet flitization based CT switching to achieve extremely low end-to-end latency in general-topology networks. Then, we have proposed an approximate queueing model to predict the end-to-end latency in such networks, with which low-latency management like end-to-end session admission control can be realized. The proposed model is built upon the PSBR scheduling mechanism, which is a latency-friendly flit scheduling method, and also upon a delay approximation technique for M/G/c queueing. Via extensive simulations, we have shown the accuracy of the proposed model in various network scenarios, revealing its potential as a useful analytical tool for general-topology CT switching networks.

In the future, we would like to develop a session admission control mechanism exploiting the proposed latency prediction model, to further show the efficacy of the developed queueing model in fulfilling the low-latency requirement. In addition, we plan to investigate the impact of transmission error on CT switching's flit forwarding, considering per-flit error probability and selective flit retransmissions.

REFERENCES

[1] M. Maier, M. Chowdhury, B. P. Rimal, and D. P. Van, "The tactile Internet: Vision, recent progress, and open challenges," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 138–145, May 2016.

[2] M. S. Elbamby, C. Perfecto, M. Bennis, and K. Doppler, "Toward low-latency and ultra-reliable virtual reality," *IEEE Netw.*, vol. 32, no. 2, pp. 78–84, Mar. 2018.

[3] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency critical IoT applications in 5G: Perspective on the design of radio interface and network architecture," *IEEE Commun. Mag.*, vol. 55, no. 2, pp. 70–78, Feb. 2017.

[4] A. Ghosh. (Sep. 2017). *5G mmWave Revolution and New Radio*. [Online]. Available: https://futurenetworks.ieee.org/images/files/pdf/5GmmWave_Webinar_IEEE_Nokia_09_20_2017_final.pdf

[5] N. A. Johansson, Y.-P.-E. Wang, E. Eriksson, and M. Hessler, "Radio access for ultra-reliable and low-latency 5G communications," in *Proc. IEEE Int. Conf. Commun. Workshop (ICCW)*, Jun. 2015, pp. 1184–1189.

[6] M. Simsek, A. Aijaz, M. Dohler, J. Sachs, and G. Fettweis, "The 5G-enabled tactile Internet: Applications, requirements, and architecture," in *Proc. IEEE Wireless Commun. Netw. Conf.*, Apr. 2016, pp. 1–6.

[7] D. P. Bertsekas, R. G. Gallager, and P. Humblet, *Data Network*. Upper Saddle River, NJ, USA: Prentice-Hall, 1992.

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

[9] K. Shin, S. Choi, and H. Kim, "Flit scheduling for cut-through switching: Towards near-zero end-to-end latency," *IEEE Access*, vol. 7, pp. 66369–66383, 2019.

[10] M. Ilyas and H. Mouftah, "Quasi cut-through: New hybrid switching technique for computer communication networks," *IEE E, Comput. Digit. Tech. UK*, vol. 131, no. 1, p. 1, 1984.

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

[12] T. Szigeti, C. Hattingh, R. Barton, and K. Briley. Jr., *End-to-End QoS Network Design: Quality of Service for Rich-Media & Cloud Networks*. Indianapolis, IN, USA: Cisco Press, 2013.

[13] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," IETF, Fremont, CA, USA, Tech. Rep. RFC 1633, 1994.

[14] A. Mankin, F. Baker, B. Braden, S. Bradner, M. O'Dell, A. Romanow, A. Weinrib, and L. Zhang, "Resource reservation protocol (RSVP)–version 1 applicability statement some guidelines on deployment," IETF, Fremont, CA, USA, Tech. Rep. RFC 2205, 1997.

[15] K. Nichols, S. Blake, F. Baker, and D. Black, "Definition of the differentiated services field (DS field) in the ipv4 and ipv6 headers," IETF, Fremont, CA, USA, Tech. Rep. RFC 2474, 1998.

[16] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[17] R. Kapoor, G. Porter, M. Tewari, G. M. Voelker, and A. Vahdat, "Chronos: Predictable low latency for data center applications," in *Proc. 3rd ACM Symp. Cloud Comput. (SoCC)*, Oct. 2012, p. 3.

[18] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, nos. 2–3, pp. 105–128, Feb. 2004.

[19] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, and W.-K. Su, "Myrinet: A gigabit-per-second local area network," *IEEE Microw.*, vol. 15, no. 1, pp. 29–36, Feb. 1995.

[20] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker, "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.

[21] A. Ademaj and H. Kopetz, "Time-triggered Ethernet and IEEE 1588 clock synchronization," in *Proc. IEEE Int. Symp. Precis. Clock Synchronization Meas., Control Commun.*, Oct. 2007, pp. 41–43.

[22] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Germany: Springer, 2019.

[23] L. Zhang, J. Liu, and K. Yang, "Quality of service modelling of virtualized wireless networks: A network calculus approach," *Mobile Netw. Appl.*, vol. 19, no. 4, pp. 572–582, Aug. 2014.

[24] L. Zhang, J. Yu, and X. Deng, "Modelling the guaranteed QoS for wireless sensor networks: A network calculus approach," *EURASIP JWCN*, vol. 2011, no. 1, p. 82, 2011.

[25] M. Fidler and A. Rizk, "A guide to the stochastic network calculus," *IEEE Commun. Surv. Tuts.*, vol. 17, no. 1, pp. 92–105, Jul. 2015.

[26] *Latency in Ethernet Switches*, Plexxi, Nashua, NH, USA, 2016.

[27] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[28] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu, and C.-S. Yang, "Network monitoring in software-defined networking: A review," *IEEE Syst. J.*, vol. 12, no. 4, pp. 3958–3969, Dec. 2018.

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

[30] 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.

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

[32] L. Kleinrock, *Communication Nets; Stochastic Message Flow Delay*. New York, NY, USA: Dover, 1972.

[33] M. Van Hoorn and H. Tijms, "Approximations for the waiting time distribution of the M/G/c queue," *Perform. Eval.*, vol. 2, no. 1, pp. 22–28, May 1982.

[34] N. Chee-Hock and S. Boon-He, *Queueing Modelling Fundamentals*. Hoboken, NJ, USA: Wiley, 2002.

[35] A. D. Polyanin and A. V. Manzhirov, *Handbook of Integral Equations*. Boca Raton, FL, USA: CRC Press, 2008.

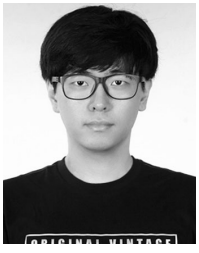
[36] R. Mittal, J. Sherry, S. Ratnasamy, and S. Shenker, "Recursively cautious congestion control," in *Proc. NSDI*, 2014, pp. 373–385.

[37] R. Sinha, C. Papadopoulos, and J. Heidemann, "Internet packet size distributions: Some observations," USC/Inf. Sci. Inst., Los Angeles, CA, USA, Tech. Rep. ISI-TR-2007-643, 2007.

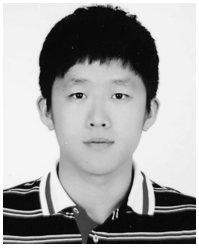
[38] V. K. Adhikari, S. Jain, and Z.-L. Zhang, "YouTube traffic dynamics and its interplay with a tier-1 ISP: An ISP perspective," in *Proc. 10th Annu. Conf. Internet Meas. (IMC)*, 2010, pp. 431–443.

[39] Cisco. (2018). *Cisco Nexus 5600 Platform 10-GBPS Switches Data Sheet*. [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-5000-series-switches/datasheet-c78-730760.html

[40] Juniper. (2019). *Ex4600 Ethernet Switch*. [Online]. Available: https://www.juniper.net/assets/de/de/local/pdf/datasheets/1000511-en.pdf



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 joint M.S. and Ph.D. (combined program) degrees with the School of Electrical and Computer Engineering. His research interests include 5G communications and low latency networking.



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 radios, super Wi-Fi, next-generation WLANs and cellular networks, and 5G communications.

HYOIL KIM (Senior Member, IEEE) 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. He was a Postdoctoral Researcher with the IBM T. 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), next-generation WLAN, 5G/6G communications, mobile cloud computing, vehicular networks, and AI-empowered networking technologies. He has served as a TPC Member of various international conferences including ACM WiNTECH 2013, the IEEE INFOCOM, in 2014 and from 2016 to 2017, the IEEE WCNC in 2016, the IEEE GLOBECOM, from 2011 to 2015 and from 2017 to 2018, the IEEE DySPAN, from 2018 to 2019, and the IEEE PIMRC, from 2015 to 2019. He is currently serving as a TPC Member of the IEEE SECON 2020 and the IEEE WCNC 2020. He has also served as the Publicity Co-Chair for ACM WiNTECH 2013. He has served as a Guest Editor for the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING (TCCN) and *Ad Hoc Networks*. He is an Area Editor of *Computer Networks* (ComNet) and an Editor of the *Journal of Communications and Networks* (JCN).