Approximate and Deployable Shortest Remaining Processing Time Scheduler

Zhiyuan Wang, Jiancheng Ye, *Member, IEEE, ACM*, Dong Lin, Yipei Chen, and John C. S. Lui, *Fellow, IEEE, ACM*

***Abstract*— The scheduling policy installed on switches of datacenters plays a significant role on congestion control. Shortest-Remaining-Processing-Time (SRPT) achieves the near-optimal average message completion time (MCT) in various scenarios, but is difficult to deploy as viewed by the industry. The reasons are two-fold: 1) many commodity switches only provide FIFO queues, and 2) the information of remaining message size is not available. Recently, the idea of emulating SRPT using only a few FIFO queues and the original message size has been coined as the approximate and deployable SRPT (ADS) design. In this paper, we provide the first theoretical study on the optimal ADS design. Specifically, we first characterize a wide range of feasible ADS scheduling policies via a unified framework, and then derive the steady-state MCT, slowdown, and impoliteness in the M/G/1 setting. Hence we formulate the optimal ADS design as a non-linear combinatorial optimization problem, which aims to minimize the average MCT given the available FIFO queues. We also take into account the proportional fairness and temporal fairness constraints based on the maximal slowdown and impoliteness, respectively. The optimal ADS design problem is NP-hard in general, and does not exhibit monotonicity or submodularity. We leverage its decomposable structure and devise an efficient algorithm to solve the optimal ADS policy. We carry out extensive flow-level simulations and packet-level experiments to evaluate the proposed optimal ADS design. Results show that the optimal ADS policy installed on eight FIFO queues is capable of emulating the true SRPT.**

***Index Terms*— Emulating SRPT, proportional and temporal fairness, scheduling policy, queueing system.**

Manuscript received May 23, 2021; revised October 31, 2021; accepted December 19, 2021; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor J.-W. Lee. Date of publication January 19, 2022; date of current version June 16, 2022. This work was supported in part by the General Research Fund (GRF) under Grant 14200321 and in part by The Chinese University of Hong Kong (CUHK) under Grant 6905407. Part of the results was presented in IEEE/ACM IWQoS 2021 [1] [DOI: 10.1109/IWQOS52092.2021.9521259]. *(Corresponding author: Jiancheng Ye.)*

Zhiyuan Wang is with the School of Computer Science and Engineering, Beihang University, Beijing 100191, China (e-mail: zhiyuanwang@buaa.edu.cn).

Jiancheng Ye and Dong Lin are with the Network Technology Laboratory and the Hong Kong Research Center, Huawei Technologies Company Ltd., Hong Kong (e-mail: yejiancheng@huawei.com; lin.dong@huawei.com).

Yipei Chen was with the Hong Kong Research Center, Huawei Technologies Company Ltd., Hong Kong. He is now with the Department of Mathematics, The Hong Kong University of Science and Technology, Hong Kong (e-mail: ychendh@connect.ust.hk).

John C. S. Lui is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: cslui@cse.cuhk.edu.hk).

Digital Object Identifier 10.1109/TNET.2022.3142148

I. INTRODUCTION

A. Background and Motivation

CONGESTION control (CC) is one of the most critical issues in the modern data-center network (DCN) [2]. To maintain reliability and scalability, most CC schemes coordinate in a distributed way, and part of the CC mechanism is implemented in the network switches [3]. Hence, the end-to-end latency depends primarily on the scheduling policy installed on these switches. Nowadays, many datacenter applications are using request-response protocols, which generate a lot of short messages. The application message is a block of packets transmitted from a sender to the receiver. In general, the message-size-based scheduling policy that prioritizes the short messages is believed to significantly reduce the average message completion time (MCT). Moreover, Shortest-Remaining-Processing-Time (`SRPT`) is known to achieve the near-optimal average completion time [4], by prioritizing jobs with the shortest *remaining* service time. However, `SRPT` is not readily deployable as viewed by the industry for two reasons:

- First, many commodity switches by default only provide support for FIFO queues per outgoing port, thus naturally enforce the First-Come-First-Serve (`FCFS`) discipline.
- Second, the *remaining* message size information needed for `SRPT` is not available in today's transport protocols.

There have been some studies (e.g., [5]–[11]) on how to harness the advantage of `SRPT` via feasible industry solutions. A promising approach is to emulate `SRPT` based on the preemptive size-based scheduling policies. The feasibility of this idea relies on the following two facts.

- First, most commodity switches provide support for *multiple* FIFO queues (typically eight to ten [6], [12], [13]).
- Second, the *original* size of a message is available to the switch.[1] This information allows the switch to prioritize the short messages in transmission.

The above two facts make it possible to emulate `SRPT` by putting shorter messages into the FIFO queue with a higher transmission priority, so that they can finish faster. This idea is coined as ***approximate and deployable SRPT (ADS)*** by Mushtaq *et al.* in [11]. Fig. 1 provides an illustration based on the switch with K FIFO queues. It works as follows:

- The switch assigns the incoming message based on its original message sizes to one of the K FIFO queues.

[1]RDMA is completely message-orientated [14]. The sender must specify the size information in the first packet to be transmitted [9].

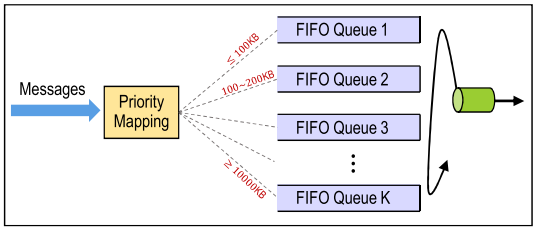


Fig. 1. An illustrative ADS design with K FIFO queues.

Each FIFO queue has a predefined size interval. In Fig 1, a smaller queue index means a higher priority for transmission. These size intervals define the size-based priority mapping.

- The switch will transmit the messages based on the predefined priority. A newly-arrived short message (e.g., assigned to FIFO queue 2) can preempt the long message being transmitted (e.g., in FIFO queue 3). The switch will resume transmitting the interrupted message when the FIFO queues with higher priority (than it) are empty.

Mushtaq *et al.* in [11] explore the ADS design space in different dimensions based on the packet-level simulation (see Section 3.1 in [11]). This paper provides the first theoretical study on two critical dimensions in ADS design, i.e., the priority mapping and the number of used FIFO queues. Despite some system-level ADS studies (e.g., [5]–[11]), there is no theoretical framework on how to jointly optimize the size-based priority mapping and the number of adopted priority levels. In this paper, we take the initial step to fill this void, and propose a unified theoretical framework for designing ADS scheduling policy. Specifically, we aim to address the following two fundamental questions:

Question 1: Given $K \geq 2$ FIFO queues in the switch, what is the optimal size-based priority mapping scheme?

Question 2: If the switch could provide sufficient FIFO queues, what is the optimal number of priority levels?

Question 1 corresponds to the practical ADS scheduling policy design given the switch with a fixed number of FIFO queues. For Question 2, we investigate the optimal number of priority levels to adopt. Although more available FIFO queues provide a better design flexibility, we show that the optimal ADS scheduling policy is not necessary to utilize all the available FIFO queues. The number of used FIFO queues may depend on factors such as the message size distribution and the network load intensity. By investigating Question 2, we reveal when it is necessary to upgrade the current switch by increasing the embedded FIFO queues.

B. Main Results and Key Contributions

Our main results and key contributions in this paper are summarized as follows:

- *A Unified Framework for ADS Scheduler:* We leverage the discrete message size, and characterize a wide range of ADS scheduling policies via a unified framework. The scheduling policies within this framework only require the original message size and a few FIFO queues, thus provide a feasible industry solution for commodity switches. Under this framework, we derive the steady-state MCT, slowdown, and impoliteness in the M/G/1 setting, which facilitates the subsequent optimal ADS policy design.
- *Problem Formulation for ADS Policy Design:* We formulate the optimal ADS design as a non-linear combinatorial optimization problem, with the goal to minimize the average steady-state MCT given the available FIFO queues in the switch. We also take into account the proportional fairness and temporal fairness constraints based on the maximal steady-state slowdown and impoliteness, respectively. The two fairness considerations enable us to avoid the starvation of long messages and control the out-of-order delivery, respectively.
- *Optimal ADS Policy:* The above ADS design problem is NP-hard in general, and does not exhibit monotonicity or sub-modularity. We leverage the decomposable structure in this problem, and devise an efficient algorithm to solve the optimal ADS policy. Our approach leads to the optimal priority mapping, and can also unveil the optimal number of priority levels. As far as we know, we are the first to jointly address the two fundamental challenges in ADS design.
- *Flow-Level Simulations:* We carry out flow-level simulations based on the realistic heavy-tail message size distributions. Results show that the optimal ADS policy on eight FIFO queues can maintain almost the same performance as SRPT in the M/G/1 setting. We also demonstrate its robustness from two aspects. First, the optimal number of priority levels is not sensitive to the load and the message size variation. Second, the optimal ADS policy derived from the M/G/1 analytical model can still retain the performance of SRPT in the non-Poisson scenario with the same load.
- *Packet-Level Experiments:* We carry out extensive packet-level experiments on NS-3. The results validate that the optimal ADS policy on eight FIFO queues is capable of emulating the true SRPT in terms of the average slowdown across different percentiles for short messages.

The remainder of this paper is organized as follows. Section II reviews the related literature. Section III introduces the system model and formulates the ADS design problem. Section IV derives the optimal ADS policy. We provide the flow-level and packet-level simulation results in Section V and Section VI, respectively. Section VII concludes this paper.

II. Literature Review

We first review some ADS implementations in Section II-A, and then compare the analytical models for ADS design in Section II-B.

A. ADS Implementation

We introduce some ADS implementations, focusing on their priority mapping schemes and the number of priority levels.

1) Priority Mapping Scheme: The priority mapping scheme usually uses either static priority or dynamic priority.

- *Static Priority based on Original Size:* Some ADS implementations define a static scheduling priority based on the original message size, thus are easy to implement. Cisco's dynamic packet prioritization (DPP) [5] adopts

two priority queues and assigns messages by comparing their original sizes to a predefined size threshold.

- *Static Priority based on Remaining Size:* Some proposals define the static priority based on the remaining size, which can achieve the near-optimal performance at the cost of a higher implementation complexity. In pFabric [6], each packet carries the number of currently remaining bytes in this flow, so that the switch can prioritize those packets with the lowest remaining size. It relies on some specialized hardwares, thus cannot be easily implemented on commodity switches. PIAS [7] provides a different technique and assumes that the flows are short until proven long. As the flow progresses, the priority decreases accordingly.
- *Dynamic Priority:* Different from pFabric and PIAS that compute the priorities at senders, pHost [8] and Homa [9] compute the priorities entirely at receivers. In this case, both pHost and Homa will assign the priorities dynamically on receivers and integrate the priorities with a receiver-driven flow control mechanism.

In this paper, we focus on the static priority defined based on the original message size, which is easy to be implemented on the commodity switches.

2) Number of Adopted Priority Levels: The aforementioned studies (i.e., [5]–[9]) usually consider a small number of FIFO queues (e.g., from 2 to 8). Moreover, Lu *et al.* in [10] focus on achieving good MCT performance using only two FIFO queues. So far, there is no formal study on the optimal number of priority levels to be adopted in ADS design. This aspect is critical in practice, since it unveils the necessity of upgrading the switch hardware. We will address this issue and propose a unified framework for ADS policy in this paper. This framework allows us to jointly optimize the priority mapping and the number of priority level, which will be elaborated in the following.

B. Analytic Model for ADS Design

Some previous studies (e.g., [6], [7]) introduce their analytical models for priority mapping optimization. These models are usually based on the K-class polling system [15], where the incoming messages are classified based on $K-1$ size thresholds. Under this model, however, it is usually challenging to optimize the ADS scheduling policy due to the non-convexity and the dimensionality:

- *Non-Convexity:* The optimization problem is usually non-convex in the $K-1$ size thresholds (i.e., the decision variables), thus it is challenging to obtain the optimal priority mapping scheme. Some studies (e.g., [6], [11]) adopt a simple equal-splitting heuristic, which has no optimality guarantee. Hence the previous studies do not systematically address Question 1.
- *Dimensionality:* Under the K-class polling system, it is not difficult to analyze the steady-state performance, but it is challenging to investigate whether K priority levels are optimal or not. Hence the aforementioned studies usually overlook the optimal number of priority levels, i.e., Question 2.

In this paper, we will propose a novel analytical model to characterize the ADS scheduling policy by leveraging the discrete message sizes. Specifically, we associate each message size with a binary variable indicating its preemption allowance. This enables us to capture a wide range of feasible ADS scheduling policies via a unified framework (as a binary vector). We then formulate the optimal ADS policy design as a combinatorial optimization problem. Although it is not monotonic or sub-modular, we derive the optimal ADS policy relying on the decomposable structure. Moreover, our approach can also unveil the optimal number of priority levels to be adopted, thus resolves the two fundamental key questions proposed in Section I-A.

III. System Model

We study the ADS scheduling policy design installed on the switch of DCN. We model this problem based on the M/G/1 queueing system. Specifically, the messages of different sizes arrive at the switch according to a Poisson process with the rate λ. We quantify the message size in the number of packets, and model it as a discrete random variable on the support set $\mathcal{N}=\{1,2,\ldots,N\}$, where N indicates the maximal message size. Let $f(n)$ denote the probability mass function (PMF), and let $F(n)\triangleq\sum_{i=1}^{n}f(i)$ denote the cumulative distribution function (CDF). Without loss of generality, we consider a normalized bandwidth, thus the service time of transmitting a size-n message is n. Accordingly, we follow some classic notations from [16] and denote the expected service time as

$$\frac{1}{\mu}\triangleq\sum_{n=1}^{N}nf(n), \tag{1}$$

where μ represents the serving rate, and the network load is $\rho\triangleq\frac{\lambda}{\mu}$.

We will introduce some preliminary results in Section III-A, and then characterize the ADS scheduling policy via a unified framework in Section III-B. Finally, we formulate the ADS design problem in Section III-C.

A. Preliminary Results

The scheduling policies in the queueing system can be evaluated according to different performance metrics. In our ADS design framework, we will focus on three crucial metrics: response time, slowdown, and impoliteness.

- Response time indicates the *efficiency* of a scheduling policy. In the network applications, the message completion time (MCT) measures the time from when the first packet of a message is sent until the last packet is received [17]. Therefore, when we focus on a single switch, MCT and response time are equivalent.
- Slowdown is used to evaluate *proportional fairness*, which is proposed in [18] and generalized in [19]. Specifically, the slowdown of a message is defined as the response time of the message divided by the time that the same message would take to complete if it was the only message in the system (e.g., [6]–[9]).
- Impoliteness is used to evaluate *temporal fairness*, which is proposed in [20]. The impoliteness perceived by a message is defined as the fraction of response time during

TABLE I
COMPARISON BETWEEN ADS FRAMEWORK AND TYPICAL SCHEDULING POLICIES

Policy	Description of the scheduling policies	Property	Deployable on FIFO Queues?	Size Information
FCFS	Transmit messages in the order they arrive	Temporally fair	Yes (one is enough)	Not need
PSJF	Preemptively transmit messages of the smallest original size	Small MCT	Yes (but requires many)	Original size
SRPT	Preemptively transmit messages of the shortest remaining size	Minimal MCT	No	Remaining size
PS	Transmit all messages simultaneously, at the same rate	Proportionally fair	\	\
PLCFS	Preemptively transmit the most recent arrival			
ADS	*Harness the advantage of* SRPT *under the tunable fairness guarantee*		Yes (Only a few)	Original size

which its seniority (i.e., the arrival order) is violated. Moreover, the impoliteness takes values in $[0, \rho]$, where ρ is the network load.

For any scheduling policy $\mathcal{P}$, we let $T_n(\mathcal{P})$, $S_n(\mathcal{P})$, and $R_n(\mathcal{P})$ denote the *steady-state* MCT, slowdown, and impoliteness associated with the size-n message, respectively. Particularly, when we focus on the normalized bandwidth, we have $S_n(\mathcal{P}) = T_n(\mathcal{P})/n$ for any $n \in \mathcal{N}$.

In Section III-B, we will introduce our proposed ADS framework. To facilitate our later discussion, we first elaborate the relation between several well-known scheduling policies and the ADS framework based on Fig. 2 and Table I.

- The scheduling policy that treats all the arrived messages equivalently is called the symmetric policy, which is always *proportionally fair*. Processor-Sharing (PS) and Preemptive-Last-Come-First-Serve (PLCFS) are two typical examples of symmetric policies, and achieve $S_n(\texttt{PS}) = S_n(\texttt{PLCFS}) = \frac{1}{1-\rho}$ for any $n \in \mathcal{N}$.
- The scheduling policy that prioritizes the short messages can reduce the average MCT. Shortest-Remaining-Processing-Time (SRPT) achieves the nearly optimal average MCT. Preemptive-Shortest-Job-First (PSJF) prioritizes the messages with the small original size. In the M/G/1 setting, PSJF is 1.5-competitive to SRPT in terms of the average MCT [20].
- The scheduling policy that prioritizes the message seniority (i.e., the arrival order) can reduce the average impoliteness, thus advocates the *temporal fairness*. It is known that First-Come-First-Serve (FCFS) is the mostly temporally fair, i.e., $R_n(\texttt{FCFS}) = 0$ for any $n \in \mathcal{N}$. However, Preemptive-Last-Come-First-Server (PLCFS) is the worst temporally fair, i.e., $R_n(\texttt{PLCFS}) = \rho$ for any $n \in \mathcal{N}$.

As we will see in Section III-B, our proposed ADS framework can generalize a wide range of preemptive size-based scheduling policies, which are shown by the *green region* in Fig. 2. The scheduling policies in this framework only rely on a few FIFO queues and the original message size information. As we will see in Section III-C, by appropriately optimizing the ADS policy within this framework, we are able to harness the advantage of SRPT under a tunable fairness guarantee.

B. Unified Framework for ADS Scheduling Policy

Recall that the ADS design aims to emulate SRPT via the priority-based scheduling installed on a few FIFO queues. The priority should only depend on the original message size instead of the remaining message size (which is not available). Therefore, we focus our ADS design on the preemptive size-based scheduling policies, and propose a unified framework to characterize them in the following. Table II summarizes the key notations.

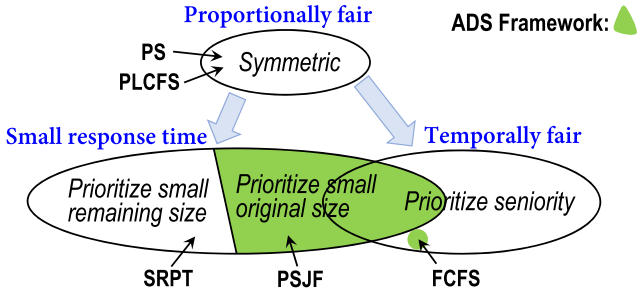


Fig. 2. Comparing some scheduling polices (e.g., PS, PLCFS, SRPT, PSJF, FCFS) to our ADS framework (i.e., green region).

1) General Characterization: Our proposed ADS policy characterization uses a binary variable $x_n \in \{0, 1\}$ to indicate the preemption allowance associated with the message size $n \in \mathcal{N}$. The physical meaning of $x_n \in \{0, 1\}$ is as follows:

- The case of $x_n = 0$ represents that the size-n messages have the same priority as the messages of size $n + 1$. In this case, all the messages of size n and size $n+1$ will be assigned to the same FIFO queue in the switch, thus will be transmitted according to the FCFS discipline.
- The case of $x_n = 1$ represents that message size n has a higher priority than other message sizes greater than n. It has two-fold implications. First, any size-n message will be assigned to FIFO queues with a higher priority than the messages greater than n. Second, a newly-arrived size-n message can immediately preempt the current transmission of a message greater than n in the switch.

Based on the above discussion, we characterize a wide range of preemptive size-based scheduling policies as the following N-dimensional binary vector

$$\boldsymbol{x} \triangleq (x_n \in \{0, 1\} \colon \forall n \in \mathcal{N}) \,. \tag{2}$$

We say that the message size $n \in \mathcal{N}$ is a *preemption point* if and only if $x_n = 1$. In particular, $x_N \in \{0, 1\}$ has no effect on the scheduling policy, since N is the maximal message size. For notation simplicity, we will fix $x_N = 1$ in this paper. Accordingly, the ADS policy $\boldsymbol{x}$ is chosen from the scheduling

TABLE II
KEY NOTATIONS

Symbol	Physical Meaning
$\mathcal{N}$	The message size set $\mathcal{N} = \{1, 2, ..., N\}$.
$f(\cdot)$	The PMF of message size distribution.
$F(\cdot)$	The CDF of message size distribution.
$\mathcal{P}$	A generic scheduling policy.
$T_n(\mathcal{P})$	Steady-state MCT of size-n message under policy $\mathcal{P}$.
$S_n(\mathcal{P})$	Steady-state slowdown of size-n message under policy $\mathcal{P}$.
$R_n(\mathcal{P})$	Steady-state impoliteness of size-n message under policy $\mathcal{P}$.
x_n	Preemption design for the message size $n \in \mathcal{N}$.
$\boldsymbol{x}$	The general ADS policy characterization, defined in (2).
$\mathcal{X}$	The ADS policy set, defined in (3).
$l_n(\boldsymbol{x})$	The largest preemption point in $\{1, 2, ..., n-1\}$, i.e., (4).
$r_n(\boldsymbol{x})$	The smallest preemption point in $\{n, n+1, ..., N\}$, i.e., (5).
$K_n(\boldsymbol{x})$	Priority level of size $n \in \mathcal{N}$ under ADS policy $\boldsymbol{x}$, i.e., (6).
K	Number of available FIFO queues in ADS design, i.e., (8).
$S_{\max}$	Maximal slowdown in ADS design, i.e., (9).
$R_{\max}$	Maximal impoliteness in ADS design, i.e., (10).
$\boldsymbol{x}^\star$	The optimal ADS policy in Problem 1.
$\mathcal{A}(i)$	Auxiliary system parameterized by i, i.e., Definition 1.
$\rho(i)$	Load of the auxiliary system $\mathcal{A}(i)$.
$V(i)$	Expected remaining service time of job being served.
$W(i)$	Expected remaining service time of all the jobs.
$H(k, i)$	The minimal value of the type-(k, i) sub-problem, i.e., (20).

policy set $\mathcal{X}$, i.e.,

$$\mathcal{X} \triangleq \left(\boldsymbol{x} \in \{0,1\}^N \colon x_N = 1\right). \tag{3}$$

The operator of the DCN could determine the adopted ADS policy $\boldsymbol{x} \in \mathcal{X}$ for the switch. Note that any scheduling policy $\boldsymbol{x} \in \mathcal{X}$ relies only on the original message size information, thus is deployable via FIFO queues in the switch. However, the policies in the set $\mathcal{X}$ differ in the number of required FIFO queues, which represents the hardware resource requirement on the switch. To better understand, we discuss two special cases in the following.

2) Two Special Cases: As shown in Fig. 2, the vector $\boldsymbol{x} \in \mathcal{X}$ generalizes PSJF and FCFS as special cases.

- The case of $\boldsymbol{x} = \mathbf{1}_N$ corresponds to Preemptive-Shortest-Job-First (PSJF), where $\mathbf{1}_N$ denotes the N-dimensional all-one vector. It strictly prioritizes the messages of the smallest original size. Given the message size set $\mathcal{N}$, it requires a total of N FIFO queues to implement PSJF on the switch, one for each message size. Hence PSJF is the most costly scheduling policy among the policy set $\mathcal{X}$.
- The case of $\boldsymbol{x} = (\mathbf{0}_{N-1}, 1)$ corresponds to First-Come-First-Serve (FCFS), which has no prioritization across all the message sizes. It only requires one FIFO queue to implement FCFS on the switch.

The above discussions indicate that PSJF requires much more FIFO queues than FCFS. Moreover, PSJF is also able to attain a smaller average MCT than FCFS. Nevertheless, we will show later that such a costly policy may not be the optimal ADS design. That is, it is possible for us to devise an ADS policy in the set $\mathcal{X}$ that achieves a smaller average MCT using much fewer FIFO queues than PSJF. For this goal, we need to characterize the message assignment outcome induced by the policy $\boldsymbol{x} \in \mathcal{X}$.

TABLE III
AN ILLUSTRATION BASED ON EXAMPLE 1

Message Size n	1	2	3	4	5	6	7	8	9
$\boldsymbol{x} = (x_n : \forall n \in \mathcal{N})$	0	0	1	0	0	0	1	0	1
$l_n(\boldsymbol{x})$	0	0	0	3	3	3	3	7	7
$r_n(\boldsymbol{x})$	3	3	3	7	7	7	7	9	9

3) Message Assignment to FIFO Queues: The scheduling priority defined by the N-dimensional vector $\boldsymbol{x} \in \mathcal{X}$ determines a unique message assignment to the FIFO queues. To understand the message assignment, we use a simple example to illustrate the connection between the binary vector $\boldsymbol{x}$ and the FIFO queues.

Example 1: Suppose that $N = 9$, then the policy $\boldsymbol{x} = (0, 0, 1, 0, 0, 0, 1, 0, 1)$ requires a total of three FIFO queues.

- *The messages of sizes $\{1, 2, 3\}$ will be assigned to FIFO queue 1, which has the highest priority.*
- *The messages of sizes $\{4, 5, 6, 7\}$ will be assigned to FIFO queue 2, which has the second highest priority.*
- *The messages of sizes $\{8, 9\}$ will be assigned to FIFO queue 3, which has the lowest priority.*

The above example shows that the scheduling policy $\boldsymbol{x} \in \mathcal{X}$ requires a total of $\sum_{n=1}^{N} x_n$ FIFO queues. To characterize the message assignment to the $\sum_{n=1}^{N} x_n$ FIFO queues, we first need to define some intermediate notations. Given the policy $\boldsymbol{x} \in \mathcal{X}$, we let $l_n(\boldsymbol{x})$ denote the largest preemption point in the message size subset $\{1, 2, \ldots, n-1\}$, i.e.,

$$\begin{aligned} l_n(\boldsymbol{x}) \triangleq \max_{1 \le i < n} \quad & i \cdot x_i \\ \textit{s.t.} \quad & x_i = 1, \end{aligned} \tag{4}$$

where we let $l_n(\boldsymbol{x}) = 0$ if $x_i = 0$ for any $i \in \{1, 2, \ldots, n-1\}$. Furthermore, let $r_n(\boldsymbol{x})$ denote the smallest preemption point in the message size subset $\{n, n+1, \ldots, N\}$, i.e.,

$$\begin{aligned} r_n(\boldsymbol{x}) \triangleq \min_{n \le i \le N} \quad & i \cdot x_i \\ \textit{s.t.} \quad & x_i = 1, \end{aligned} \tag{5}$$

where we have $r_n(\boldsymbol{x}) = n$ if $x_n = 1$.

The above definitions imply $l_n(\boldsymbol{x}) < n \le r_n(\boldsymbol{x})$ for any size $n \in \mathcal{N}$. Table III illustrates $l_n(\boldsymbol{x})$ and $r_n(\boldsymbol{x})$ based on Example 1. Moreover, Proposition 1 presents the general message assignment outcome induced by the policy $\boldsymbol{x} \in \mathcal{X}$ based on $l_n(\boldsymbol{x})$ and $r_n(\boldsymbol{x})$. The proof is given in appendix.

Proposition 1: For any $n \in \mathcal{N}$, the messages of sizes in the set $\{l_n(\boldsymbol{x}) + 1, l_n(\boldsymbol{x}) + 2, \ldots, r_n(\boldsymbol{x})\}$ will be assigned to the FIFO queue with the $K_n(\boldsymbol{x})$-th priority, where $K_n(\boldsymbol{x})$ is

$$K_n(\boldsymbol{x}) \triangleq \sum_{i=1}^{r_n(\boldsymbol{x})} x_i. \tag{6}$$

So far, we have introduced the general ADS framework and the corresponding message assignment outcome. Next we will formulate the optimal ADS design problem.

C. Problem Formulation

We will formulate the optimal ADS scheduling policy design based on the steady-state performance metrics. Following the notations in Section III-A, we let $T_n(\boldsymbol{x})$, $S_n(\boldsymbol{x})$, and $R_n(\boldsymbol{x})$ denote the steady-state MCT, slowdown, and impoliteness associated with the size-n message under the ADS policy $\boldsymbol{x} \in \mathcal{X}$. We will introduce how to derive them in Section IV-A.

To emulate SRPT, our optimal ADS design formulation aims to minimize the average steady-state MCT. Moreover, we will also take into account other crucial requirements, i.e., the number of available FIFO queues and the fairness issues. Next we introduce the details.

1) Average Steady-State MCT: Based on the above discussion, the average steady-state MCT achieved by the policy $\boldsymbol{x} \in \mathcal{X}$ is given by

$$\bar{T}(\boldsymbol{x}) \triangleq \sum_{n=1}^{N} T_n(\boldsymbol{x}) f(n), \tag{7}$$

where $f(\cdot)$ is the probability mass function of the message size distribution. We will derive $T_n(\boldsymbol{x})$ in Section IV-A.

2) Number of FIFO Queues: Let K denote the number of available FIFO queues in the switch, which is the hardware limitation in practice. Hence a feasible ADS policy $\boldsymbol{x} \in \mathcal{X}$ should not require more than K FIFO queues, i.e.,

$$\sum_{n=1}^{N} x_n \leq K. \tag{8}$$

Particularly, the case of $K = 1$ will force the feasible scheduling policy to be FCFS, i.e., $\boldsymbol{x} = (\mathbf{0}_{N-1}, 1)$.

3) Fairness Requirement: In general, reducing the average MCT requires that the scheduling policy should prioritize the short messages. However, prioritizing short messages may lead to undesirable outcomes. This motivates us to take into account the fairness issue from the following two aspects.

- First, prioritizing short messages may result in a significantly large MCT for long messages. To prevent the starvation of long messages, we introduce the following constraints

  $$S_n(\boldsymbol{x}) \leq S_{\max}, \quad \forall n \in \mathcal{N}, \tag{9}$$

  which indicates that the *maximal steady-state slowdown* should be no greater than $S_{\max}$. According to the preliminary results in Section III-A, the case of $S_{\max} = \frac{1}{1-\rho}$ corresponds to proportional fairness (e.g., [18], [19]).
- Second, prioritizing short messages may lead to the out-of-order delivery, which could result in a significant jitter for video streaming applications [21]. To prevent the serious out-of-order delivery, the ADS scheduling policy should respect the message seniority to some degree. Hence we introduce the following constraints

  $$R_n(\boldsymbol{x}) \leq R_{\max}, \quad \forall n \in \mathcal{N}, \tag{10}$$

  which indicates that the *maximal steady-state impoliteness* should be no greater than $R_{\max}$. According to the preliminary results in Section III-A, the case of $R_{\max} = 0$ forces the policy to be FCFS, i.e., $\boldsymbol{x} = (\mathbf{0}_{N-1}, 1)$.

4) ADS Policy Design Problem: Based on the above discussions, we formulate the optimal ADS policy design in Problem 1, which aims to minimize the average MCT considering the aforementioned three types of constraints.

Problem 1 (ADS Policy Design)**.**

$$\begin{aligned} \boldsymbol{x}^{\star} \triangleq \arg\min \quad & \sum_{n=1}^{N} T_n(\boldsymbol{x}) f(n) \\ \textit{s.t.} \quad & (8), (9), (10), \\ \textit{var.} \quad & \boldsymbol{x} \in \mathcal{X}. \end{aligned} \tag{11}$$

Problem 1 is a non-linear combinatorial optimization problem, which is NP-hard in general. Moreover, it does not exhibit monotonicity or sub-modularity, thus the greedy algorithm has no performance guarantee in this problem. In Section IV, we will introduce how to efficiently solve Problem 1 by exploiting its decomposable structure.

IV. ADS Policy Design

In this section, we first derive the steady-state performance of the ADS policy $\boldsymbol{x} \in \mathcal{X}$ in Section IV-A. We then introduce the key idea of solving Problem 1 and the detailed algorithm in Section IV-B and Section IV-C, respectively.

A. Steady-State Performance

We will derive the steady-state MCT $T_n(\boldsymbol{x})$, slowdown $S_n(\boldsymbol{x})$, and impoliteness $R_n(\boldsymbol{x})$ achieved by the ADS policy $\boldsymbol{x} \in \mathcal{X}$ based on a series of auxiliary systems.

1) Auxiliary Systems: Given the network workload characteristics $\{\lambda, \mathcal{N}, f(\cdot)\}$, we define the auxiliary system $\mathcal{A}(i)$ in the following.

Definition 1: The auxiliary system $\mathcal{A}(i)$ is an M/G/1 queueing system satisfying the following conditions:

1) *The arrival rate of $\mathcal{A}(i)$ is $\lambda_A \triangleq \lambda F(i)$.*
2) *The service time distribution of $\mathcal{A}(i)$ is*

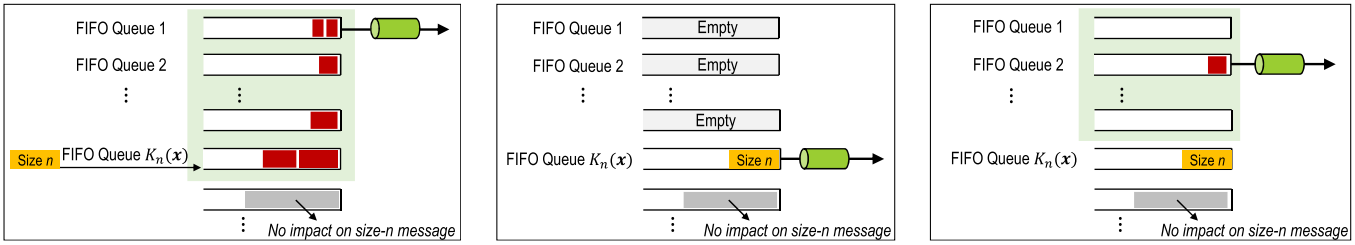
$$f_A(t) \triangleq \begin{cases} \dfrac{f(t)}{F(i)}, & \textit{if } t \in \{1, 2, \ldots, i\}, \\ 0, & \textit{otherwise}. \end{cases} \tag{12}$$

As we will see later, the steady-state analysis of the policy $\boldsymbol{x} \in \mathcal{X}$ is closely related to the above auxiliary systems. To facilitate our later discussion, let us follow some classic notations from [16], and introduce three formulas $\rho(i)$, $V(i)$, and $W(i)$ for the auxiliary system $\mathcal{A}(i)$.

- First, we let $\rho(i)$ denote the load of the auxiliary system $\mathcal{A}(i)$, i.e.,

  $$\rho(i) \triangleq \lambda_A \sum_{t=1}^{i} t f_A(t) = \lambda \sum_{t=1}^{i} t f(t). \tag{13}$$

- Second, we let $V(i)$ denote the expected remaining service time of the job being served at a random time



(a) A newly-arrived size-n message is assigned to the FIFO queue of the $K_n(\boldsymbol{x})$-th priority.

(b) The switch starts transmitting this size-n message.

(c) The size-n message may be interrupted by shorter messages with a higher priority.

Fig. 3. An illustrative transmission progress of a size-n message.

point in the auxiliary system $\mathcal{A}(i)$. Given the M/G/1 setting, we have

$$V(i) \triangleq \frac{\lambda_A}{2}\sum_{t=1}^{i} t^2 f_A(t) = \frac{\lambda}{2}\sum_{t=1}^{i} t^2 f(t). \tag{14}$$

- Third, we let $W(i)$ denote the expected remaining service time of all the jobs in the auxiliary system $\mathcal{A}(i)$ at a random time point. Based on Kleinrock's Conservation Law [22], we have

$$W(i) \triangleq \frac{V(i)}{1-\rho(i)}. \tag{15}$$

2) Steady-State MCT and Slowdown: We now derive the steady-state MCT $T_n(\boldsymbol{x})$ based on the aforementioned auxiliary systems. We will take a representative size-n message as an example, and elaborate the scheduling progress of the switch. As shown in Fig. 3, there are three major phases:

- Fig. 3(a): When a size-n message (i.e., the yellow rectangle) arrives, it will be assigned to the FIFO queue of the $K_n(\boldsymbol{x})$-th priority according to Proposition 1.
- Fig. 3(b): The switch will not start transmitting this size-n message until those messages waiting in the top $K_n(\boldsymbol{x})$ FIFO queues are completed.
- Fig. 3(c): When this size-n message is being transmitted, any newly-arrived shorter messages with a higher priority will immediately preempt its current transmission.

The above discussions imply that the steady-state MCT $T_n(\boldsymbol{x})$ satisfies the following equation

$$T_n(\boldsymbol{x}) = \underbrace{W\big(r_n(\boldsymbol{x})\big)}_{\text{Part I}} + \underbrace{n}_{\text{Part II}} + \underbrace{T_n(\boldsymbol{x})\rho\big(l_n(\boldsymbol{x})\big)}_{\text{Part III}}, \tag{16}$$

which implies that the steady-state MCT $T_n(\boldsymbol{x})$ equals to the summation of three parts. Solving (16) with respect to $T_n(\boldsymbol{x})$ will lead to the closed-form expression. To understand this equation, let us elaborate the physical meaning of the three parts on the right-hand-side of (16):

- Part I in (16) represents the expected remaining service time of messages in the top $K_n(\boldsymbol{x})$ FIFO queues when the size-n message arrives. We obtain this term in two steps. First, Proposition 1 indicates that the messages of sizes $\{1, 2, \ldots, r_n(\boldsymbol{x})\}$ will be assigned to top $K_n(\boldsymbol{x})$ FIFO queues. Second, the top $K_n(\boldsymbol{x})$ FIFO queues can be viewed as a subsystem (i.e., the green region in Fig. 3(a)), which is equivalent to the auxiliary system $\mathcal{A}(r_n(\boldsymbol{x}))$. Finally, substituting $r_n(\boldsymbol{x})$ into (15) yields Part I in (16).
- Part II in (16) represents the service time of transmitting the size-n message itself given the normalized bandwidth.
- Part III in (16) represents the expected service time of the messages that arrive later but have a higher priority than this size-n message during the response time $T_n(\boldsymbol{x})$. We obtain this term in two steps. First, Proposition 1 shows that the messages of sizes in $\{1, 2, \ldots, l_n(\boldsymbol{x})\}$ will be assigned to the top $K_n(\boldsymbol{x})-1$ FIFO queues, thus have a higher priority than the size-n message. Second, the top $K_n(\boldsymbol{x})-1$ FIFO queues can be viewed as a subsystem (i.e., the green region in Fig. 3(c)), which is equivalent to the auxiliary system $\mathcal{A}(l_n(\boldsymbol{x}))$. Therefore, $T_n(\boldsymbol{x})\rho(l_n(\boldsymbol{x}))$ represents the expected service time of the messages that will interrupt this size-n message during its transmission.

Based on Equation (16), we derive the steady-state MCT of the size-n message as follows:

$$T_n(\boldsymbol{x}) = \left[\frac{V(r_n(\boldsymbol{x}))}{1-\rho(r_n(\boldsymbol{x}))} + n\right]\frac{1}{1-\rho(l_n(\boldsymbol{x}))}, \tag{17}$$

where $\rho(\cdot)$ and $V(\cdot)$ are given in (13) and (14), respectively. Given the normalized bandwidth, the steady-state slowdown seen by a size-n message is:

$$S_n(\boldsymbol{x}) = \left[\frac{V(r_n(\boldsymbol{x}))}{1-\rho(r_n(\boldsymbol{x}))}\cdot\frac{1}{n} + 1\right]\frac{1}{1-\rho(l_n(\boldsymbol{x}))}. \tag{18}$$

3) Steady-State Impoliteness: We will take a representative size-n message as an example, and derive the steady-state impoliteness. Recall that the impoliteness perceived by the size-n message is defined as the fraction of its MCT during which the seniority (i.e., the arrival order) of this message is violated [20]. We will derive the steady-state impoliteness $R_n(\boldsymbol{x})$ in two steps:

- According to the elaboration on Fig. 3, the seniority of this size-n message will be violated if and only if the later-arrived messages have a higher priority than this size-n message. In this case, the current transmission of the size-n message will be interrupted, thus its serenity is not respected.
- According to the discussion on Part III in (16), $T_n(\boldsymbol{x})\rho(l_n(\boldsymbol{x}))$ represents the expected total service time of the messages that can interrupt the size-n message during its completion time $T_n(\boldsymbol{x})$.

The above two aspects indicate that the steady-state impoliteness perceived by the size-n message is given by

$$R_n(\boldsymbol{x}) \triangleq \frac{T_n(\boldsymbol{x})\rho(l_n(\boldsymbol{x}))}{T_n(\boldsymbol{x})} = \rho(l_n(\boldsymbol{x})). \tag{19}$$

Note that $R_n(\boldsymbol{x})$ is non-decreasing in n. That is, the longer messages perceive the greater impoliteness.

So far, we have derived the steady-state performance achieved by the ADS policy $\boldsymbol{x} \in \mathcal{X}$. We are now ready to analyze the optimal ADS policy design in Problem 1.

B. Key Idea of Solving Problem 1

We will introduce the key idea of solving Problem 1 based on its decomposable structure, the sub-problem definition, and the recursive relation.

1) Decomposable Structure: According to the definition of the N-dimensional binary vector $\boldsymbol{x}$, Problem 1 has a decomposable structure. Specifically, the preemption design $x_n = 1$ indicates that the messages of sizes $\{1, 2, \ldots, n\}$ have a higher priority than the messages of sizes $\{n+1, n+2, \ldots, N\}$. This naturally leads to Proposition 2. The proof is given in appendix.

Proposition 2: For any $i, n, j \in \mathcal{N}$ satisfying $i \leq n < j$, the following is true: if $x_n = 1$, then $x_j \in \{0, 1\}$ does not affect the size-i messages' steady-state MCT $T_i(\boldsymbol{x})$, slowdown $S_i(\boldsymbol{x})$, and impoliteness $R_i(\boldsymbol{x})$.

Proposition 2 shows that the preemption design $x_n = 1$ enables us to decompose the objective and constraints in Problem 1 with respect to the size index n. This allows us to decompose Problem 1 into sub-problems, and then solve the original problem in a recursive manner. Next we define the sub-problem of Problem 1.

2) Sub-Problem for ADS Design: Based on Problem 1, we will define a series of sub-problems for ADS policy design. Problem 2 corresponds to the type-(k, i) sub-problem.

Problem 2 (Sub-Problem in ADS Design)**.** *For any $k \in \{1, 2, ..., K\}$ and $i \in \{1, 2, ..., N\}$, the type-(k, i) sub-problem is as follows:*

$$\begin{aligned}
H(k,i) \triangleq \min \quad & \sum_{n=1}^{i} T_n(\boldsymbol{x}) f(n) && \text{(20a)}\\
\textit{s.t.} \quad & x_i = 1, && \text{(20b)}\\
& \sum_{n=1}^{i} x_n \leq k, && \text{(20c)}\\
& S_n(\boldsymbol{x}) \leq S_{\max}, \quad \forall n \leq i, && \text{(20d)}\\
& R_n(\boldsymbol{x}) \leq R_{\max}, \quad \forall n \leq i, && \text{(20e)}\\
\textit{var.} \quad & \{x_1, x_2, ..., x_i\} \in \{0, 1\}^i. && \text{(20f)}
\end{aligned}$$

We let $H(k, i)$ denote the optimal value of the type-(k, i) sub-problem. For presentation convenience, we will refer to $H(\cdot, \cdot)$ as the cost matrix. Note that the type-(K, N) sub-problem is mathematically equivalent to Problem 1. Therefore, the optimal ADS policy $\boldsymbol{x}^\star$ defined in Problem 1 and the cost $H(K, N)$ satisfy

$$H(K, N) = \sum_{n=1}^{N} T_n(\boldsymbol{x}^\star) f(n). \tag{21}$$

Furthermore, to understand the major rationale of the type-(k, i) sub-problem, we have two-fold elaboration:

- First, (20b) shows that the size i is a *presumed* preemption point. This is crucial to maintain the decomposable structure according to Proposition 2. In this case, the type-(k, i) sub-problem is only related to the top i binary variables, i.e., (20f), which enables us to focus on the message size subset $\{1, 2, \ldots, i\}$ in the objective function (20a) and the fairness constraints (20d) and (20e).
- Second, (20c) shows that the type-(k, i) sub-problem requires that there should be at most k preemption points in the message size subset $\{1, 2, \ldots, i\}$, where i is the presumed one.

In Section IV-C, we will derive the optimal ADS policy $\boldsymbol{x}^\star$ based on the cost matrix $H(\cdot, \cdot)$. Before, that, we need to efficiently calculate the cost matrix via the following recursive relation.

3) Recursive Relation: To facilitate the later discussion, we define some intermediate functions as follows:

$$\hat{T}_n(s, e) = \left[\frac{V(e)}{1-\rho(e)} + n\right] \frac{1}{1-\rho(s)}, \tag{22a}$$

$$\hat{S}_n(s, e) = \left[\frac{V(e)}{1-\rho(e)} \cdot \frac{1}{n} + 1\right] \frac{1}{1-\rho(s)}, \tag{22b}$$

$$\hat{R}(i) = \rho(i), \tag{22c}$$

where $\rho(\cdot)$ and $V(\cdot)$ are given in (13) and (14), respectively. Based on (22), one can express the size-n message's steady-state MCT in (17), the slowdown in (18), and the impoliteness in (19) as follows:

$$\begin{aligned}
T_n(\boldsymbol{x}) &= \hat{T}_n\big(l_n(\boldsymbol{x}), r_n(\boldsymbol{x})\big),\\
S_n(\boldsymbol{x}) &= \hat{S}_n\big(l_n(\boldsymbol{x}), r_n(\boldsymbol{x})\big),\\
R_n(\boldsymbol{x}) &= \hat{R}(l_n(\boldsymbol{x})).
\end{aligned} \tag{23}$$

Lemma 1 presents the recursive relation for the cost matrix $H(\cdot, \cdot)$. We will elaborate how it works in the following proof.

Lemma 1: The cost $H(k, i)$ in Problem 2 satisfies

$$H(k,i) = \min_{0 \leq q < i} H(k-1, q) + \sum_{n=q+1}^{i} \hat{T}_n(q, i) f(n) \tag{24a}$$

$$\textit{s.t.} \quad \hat{S}_{q+1}(q, i) \leq S_{\max}, \tag{24b}$$

$$\hat{R}(q) \leq R_{\max}, \tag{24c}$$

where $H(k-1, q)$ represents the cost of the type-$(k-1, q)$ sub-problem.

Proof of Lemma 1: To understand (24), we consider a message size $q \in \{0, 1, \ldots, i-1\}$, and then focus on the type-(k, i) sub-problem and the type-$(k-1, q)$ sub-problem. The major reasoning consists of the following three steps:

- First, the type-(k, i) sub-problem requires that there should be at most k preemption points among the message sizes $\{1, 2, \ldots, i\}$, where the message size i is a presumed one. The type-$(k-1, q)$ sub-problem requires that

there should be at most $k-1$ preemption points among the message sizes $\{1, 2, \ldots, q\}$, where the message size q is a presumed one.
- Second, given the above two presumed preemption points q and i, if there is no other preemption point in the size set $\{q+1, q+2, \ldots, i-1\}$, then for any message size $n \in \{q+1, q+2, \ldots, i\}$, the corresponding steady-state MCT, slowdown, impoliteness are $\hat{T}_n(q,i)$, $\hat{S}_n(q,i)$, and $\hat{R}(q)$, respectively.
- Third, $\hat{S}_n(q,i)$ is decreasing in $n \in \{q+1, q+2, \ldots, i\}$, the inequality (24b) is the necessary and sufficient condition of $S_n(\boldsymbol{x}) \leq S_{\max}$ for any $n \in \{q+1, q+2, \ldots, i\}$.

Based on the above discussion, appropriately choosing the optimal $q \in \{0, 1, \ldots, i-1\}$ will lead to (24). □

C. Algorithm Design

Algorithm 1 summarizes our proposed approach for solving the optimal ADS policy $\boldsymbol{x}^\star$. It works in two steps as follows:

- ***Lines 2-10*** calculate the cost $H(k,i)$ and the policy $P(k,i)$ for any $k \in \{1, 2, \ldots, K\}$ and $i \in \{1, 2, \ldots, N\}$. Each iteration (k,i) consists of two parts:
  - *Lines 4-8* utilize the recursive relation in Lemma 1 to calculate $H(k,i)$ and $P(k,i)$.
  - *Lines 9-10* obtain the cost $H(k,i) = \Psi(q^*)$ and the policy $P(k,i) = q^*$ for this iteration (k,i).
- ***Lines 11-15*** generate the optimal ADS policy $\boldsymbol{x}^\star$ based on the policy matrix $P(\cdot,\cdot)$. Specifically, we start with the type-(K,N) sub-problem in Line 11, and find out the optimal preemption points repetitively in Lines 12-15.

The running time complexity and the space complexity of Algorithm 1 are $\mathcal{O}(KN^2)$ and $\mathcal{O}(KN)$, respectively. The number of available FIFO queues K is usually a small number for commodity switch. Hence the implementation complexity of Algorithm 1 primarily depends on the number of different message sizes N in practice.[2]

Our above analysis takes into account the limitation from the number of available FIFO queues, i.e., the constraint (8). Hence we have addressed Question 1. Roughly speaking, the analysis above is based on the two-dimensional dynamic programming. To obtain the optimal number of adopted priority levels, i.e., Question 2, one should remove the constraint (8) and investigate the optimal ADS design. This setting will not increase the analytical challenge; in fact the absence of constraint (8) will simplify the sub-problem definition and the recursive relation. In this case, the problem becomes a standard one-dimensional dynamic programming. It is not difficult to show that the corresponding simplified algorithm has the running time complexity $\mathcal{O}(N^2)$ and the space complexity $\mathcal{O}(N)$. In Section V, we will investigate how the optimal number of priority levels scales as N increases.

So far, we have finished the theoretical analysis of the ADS policy design. To evaluate the performance, we first provide the flow-level simulation results in Section V. We then carry out the packet-level experiments on NS-3 [23] in Section VI.

[2]In this paper, we consider all the size values no greater than N, i.e., $\mathcal{N} = \{1, 2, \ldots, N\}$. To reduce the complexity, one could consider $\mathcal{N} = \{1, 10, 20, 30, \ldots, N\}$ as the feasible set for ADS design. In this case, the omitted sizes (e.g., {2,3,...,9,...}) will not act as the preemption points.

Algorithm 1

Input : Arrival rate λ and message size PMF $f(\cdot)$
Output: The optimal ADS policy $\boldsymbol{x}^\star$ in Problem 1

1 **Initial** $H(0,0) = 0$, $P(0,0) = 0$, and $\boldsymbol{x}^\star = (\mathbf{0}_{N-1}, 1)$
2 **for** $k = 1$ **to** K **do**
3 **for** $i = 1$ **to** N **do**
4 **for** $q = 0$ **to** $i-1$ **do**
5 **if** $\hat{S}_{q+1}(q,i) > S_{\max}$ ***or*** $\hat{R}(q) > R_{\max}$ **then**
6 $\Psi(q) = +\infty$
7 **else**
8 $\Psi(q) = H(k-1,q) + \sum_{n=q+1}^{i} \hat{T}_n(q,i) f(n)$
9 Find $q^* = \min_{0 \leq q < i} \Psi(q)$
10 Set $H(k,i) = \Psi(q^*)$ and $P(k,i) = q^*$
11 Set $k = K$ and $n = N$
12 **repeat**
13 Set $n = P(k,n)$ and $x_n^\star = 1$
14 $k = k - 1$
15 **until** $k = 0$;

V. Flow-Level Simulation

In the flow-level simulation, we first evaluate the steady-state performance of the optimal ADS design in Section V-A. We then demonstrate its robustness from the perspective of message size distribution and non-Poisson arrival in Section V-B and Section V-C, respectively.

A. Performance Comparison

We will compare the steady-state performance of different scheduling policies in the M/G/1 setting. Specifically, we follow the previous studies (e.g., [6], [11], [24]) and assume that the message size follows a realistic heavy-tail distribution, where 50% of the messages are of 1KB, 35% are between (1KB,201KB], and 15% are between (201KB,3000KB].[3] Hence we have $N = 3000$ in this setting. As for the scheduling policies, we will evaluate the following cases:

- Case $\texttt{Alg}(K)$ denotes the optimal ADS policy $\boldsymbol{x}^\star$ generated by Algorithm 1, which requires at most K FIFO queues due to the constraint (8). In particular, the case $\texttt{Alg}(1)$ is equivalent to `FCFS`.
- Case `Alg` denotes the optimal ADS policy without the constraint (8). Hence `Alg` may require up to N FIFO queues, which is unknown in advance.
- Case $\texttt{ES}(K)$ denotes the simple equal-splitting heuristic installed on K FIFO queues. This heuristic is widely used in some previous studies (e.g., [6], [7], [11]), thus it is a reasonable baseline.

[3]The message sizes among the intervals (1KB,201KB] and (201KB,3000KB] are uniformly distributed.

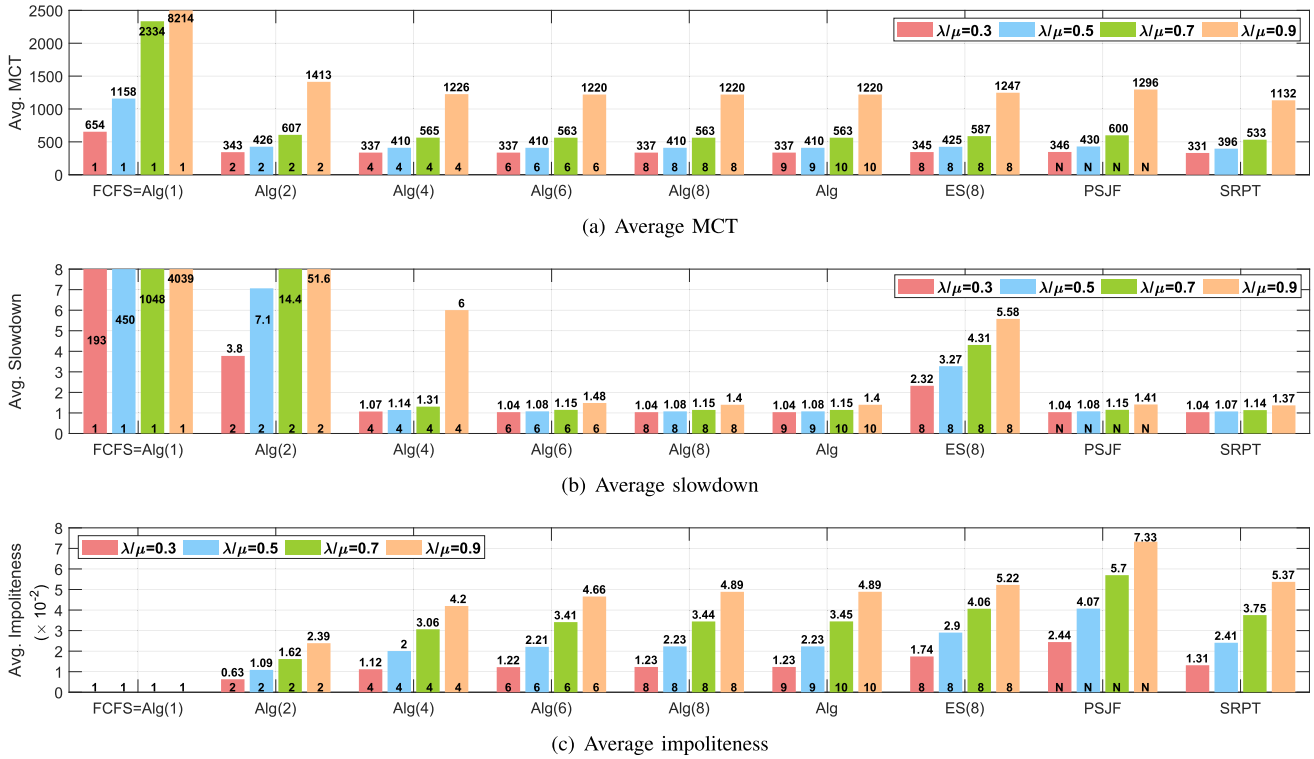


Fig. 4. Steady-state performance (The number of required FIFO queues is labeled at each bar bottom).

We will compare the above cases with PSJF and SRPT. Note that PSJF and SRPT do not have fairness guarantee, while our optimal ADS policy has a tunable trade-off given the constraints (9) and (10). In this simulation, we will set $S_{\max}$ and $R_{\max}$ to large values for the cases $\texttt{Alg}(K)$ and Alg. This enables us to make the reasonable performance comparison with PSJF and SRPT.

Fig. 4 plots the average steady-state MCT, slowdown, and impoliteness. In each sub-figure, the four bars for each policy correspond to different load $\lambda/\mu \in \{0.3, 0.5, 0.7, 0.9\}$. At the bottom of each bar, we label the number of required FIFO queues. At the top of each bar, we label the average steady-state performance. Next we elaborate the major observations obtained from Fig. 4.

Observation 1 Let us first focus on cases $\texttt{Alg}(K)$ for $K \in \{1, 2, 4, 6, 8\}$, i.e., the left five cases in each sub-figure. Given the same load, Fig. 4(a) shows that the average MCT has a significant drop from $K = 1$ to $K = 2$, while the average MCT only slightly decreases in $K \in \{2, 4, 6, 8\}$. This is because that the average MCT is dominated by the contribution from just a few long messages [25], thus is not informative when there are many short messages. Fig. 4(b) shows that the average slowdown of $\texttt{Alg}(K)$ has a significant drop from $K = 2$ to $K = 4$. Moreover, under the heavy load 0.9, the average slowdown still has an evident drop from $K = 4$ to $K = 6$. This means that the optimal two-priority policy may be good enough for reducing the average MCT, but is not good enough for slowdown. Fig. 4(c) shows that the average impoliteness of $\texttt{Alg}(K)$ monotonically increases in $K \in \{1, 2, 4, 6, 8\}$. That is, the optimal ADS policy with more priority levels will reduce the temporal fairness. Furthermore,

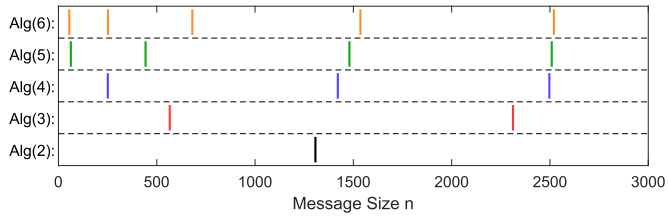
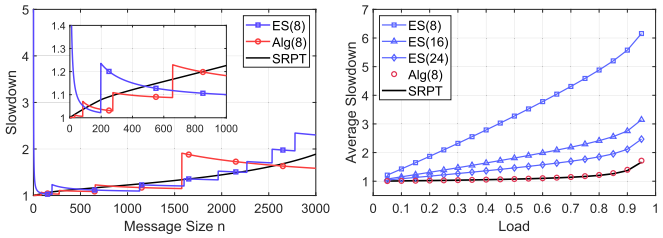


Fig. 5. Structure of $\boldsymbol{x}^\star$ in $\texttt{Alg}(K)$ with $\lambda/\mu = 0.9$.

we use Fig. 5 to show the structure of the optimal ADS policy $\boldsymbol{x}^\star$ in $\texttt{Alg}(K)$ where $K \in \{2, 3, 4, 5, 6\}$. Specifically, the horizontal axis represents the message size. The vertical lines represent the preemption points (i.e., $\{i \in \mathcal{N} : x_i^\star = 1\}$) in the optimal ADS policy $\boldsymbol{x}^\star$.

Observation 2: Let us focus on cases $\texttt{Alg}(8)$, SRPT and PSJF. First, comparing $\texttt{Alg}(8)$ and SRPT in Fig. 4(a) and Fig. 4(b), we find that the optimal ADS policy installed on eight FIFO queues is very close to the true SRPT in terms of the average MCT and slowdown. Moreover, Fig. 4(c) shows that $\texttt{Alg}(8)$ is more polite than SRPT. Second, comparing $\texttt{Alg}(8)$ and PSJF in each sub-figure shows that $\texttt{Alg}(8)$ outperforms PSJF (that requires $N = 3000$ FIFO queues) in terms of reducing the average MCT, slowdown, and impoliteness.

Observation 3: Let us focus on cases $\texttt{Alg}(8)$ and $\texttt{ES}(8)$. Fig. 4(a) and Fig. 4(c) show that $\texttt{Alg}(8)$ slightly outperform $\texttt{ES}(8)$ in terms of average MCT and impoliteness, respectively. Fig. 4(b) shows that $\texttt{Alg}(8)$ significantly outperforms $\texttt{ES}(8)$ in terms of the average slowdown. To better understand, we further plot the steady-state slowdown for each message



(a) Slowdown at load 0.5 (b) $\texttt{Alg}(8)$ vs $\texttt{ES}(K)$

Fig. 6. Compare $\texttt{ES}(K)$ and $\texttt{Alg}(K)$.

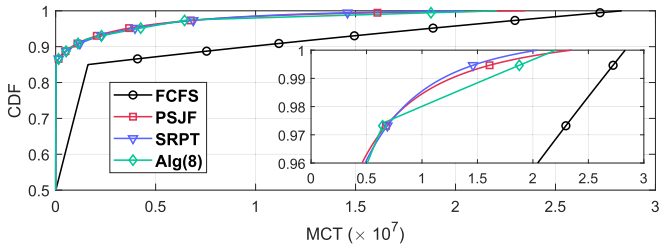


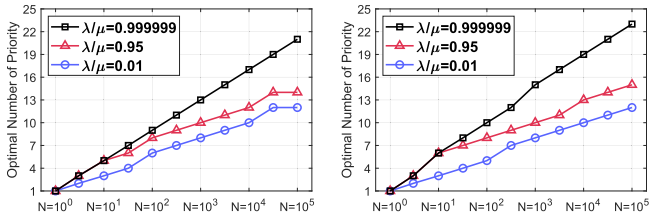
Fig. 7. MCT distribution under $\lambda/\mu = 0.9$.

size $n \in \mathcal{N}$ under load 0.5 in Fig. 6(a), where the three curves correspond to SRPT, $\texttt{Alg}(8)$ and $\texttt{ES}(8)$, respectively. Fig. 6(a) shows that $\texttt{Alg}(8)$ performs much better than $\texttt{ES}(8)$ in terms of emulating SRPT for short messages (e.g., smaller than 200KB). Furthermore, Fig. 6(b) compares the average slowdown of $\texttt{Alg}(8)$ and $\texttt{ES}(K)$ under different load, which indicates that the equal-splitting heuristic under 24 FIFO queues is still not as good as $\texttt{Alg}(8)$.

Observation 4: Let us focus on the case Alg itself. In Fig. 4, the number of required FIFO queues is labeled at the bar bottom. It shows that the optimal number of priority levels in case Alg are $\{9, 9, 10, 10\}$ under the load $\{0.3, 0.5, 0.7, 0.9\}$, respectively. This has two-fold implications. First, the optimal number of priority levels is much smaller than the number of different message sizes $N = 3000$ in this simulation setup. Second, a heavier load may lead to more priority levels in the optimal ADS design. In Section V-B, we will further investigate how the network load and message size distribution affect the optimal number of priority levels in case Alg.

So far, we have focused on the average results. Now let us investigate the percentile MCT shown in Fig. 7. Overall, $\texttt{Alg}(8)$ performs similar to SRPT and PSJF. In terms of the 99-th percentile MCT, however, $\texttt{Alg}(8)$ performs slightly worse than SRPT and PSJF. The reasons are two-fold:

- Our ADS design only relies on the FIFO queues to emulate SRPT, which leads to the drawback on the 99-th percentile MCT.
- PSJF is a special case in our design space $\mathcal{X}$. The goal of our ADS design is to reduce the average MCT, which possibly scarifies the 99-th percentile MCT. This is the reason that $\texttt{Alg}(8)$ performs better than PSJF in terms of the average MCT, but could be slightly worse off than PSJF in terms of the 99-th percentile MCT.



(a) Uniform distribution (b) Pareto distribution

Fig. 8. Impact of N on case Alg.

B. Optimal Number of Priority Levels

Next we will focus on the case Alg, i.e., the optimal ADS policy without the constraint (8). To unveil the impact of N, we follow the previous studies (e.g., [11], [26]) and consider the following two heavy-tail message size distributions:

- In the first scenario, we suppose that the message size is uniformly distributed on the support set $\{1, 2, \ldots, N\}$.
- In the second scenario, we suppose that the message size follows the Pareto distribution truncated on the support set $\{1, 2, \ldots, N\}$. The scale and shape parameters are 1 and 0.0001, respectively.

Fig. 8 plots the optimal number of adopted priority levels in the above two scenarios. In each sub-figure, the horizontal axis represents N in the *logarithmic* scale, and the vertical axis represents the optimal number of priority levels in the case Alg. The three curves in each sub-figure represent different load. The two sub-figures in Fig. 8 lead to the consistent insights, which are two-fold:

- First, the circle curve and triangle curve in each sub-figure of Fig. 8 show that a significant load increase (i.e., from 0.01 to 0.95) increases only a few (i.e., no more than three) priority levels in the optimal ADS design. That is, the number of FIFO queues required by the optimal ADS design is *not sensitive* to the load variation.
- Second, the optimal number of priority levels increases in N according to the order $\mathcal{O}(\log N)$. That is, the optimal number of priority levels is *much smaller* than the number of different message sizes (or the message size variation).

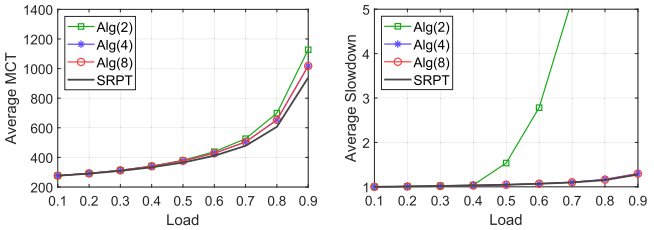
Therefore, the optimal ADS design is robust in the sense that a tremendous environment change only slightly affects its hardware requirement, i.e., the number of FIFO queues.

C. Non-Poisson Arrival

Recall that our theoretical analysis in Section III and Section IV is based on the M/G/1 queueing model. Next we evaluate the impact of the non-Poisson arrival on the optimal ADS design. Specifically, we will focus on the realistic message size distribution used in Section V-A, and consider the following two non-Poisson arrival scenarios:

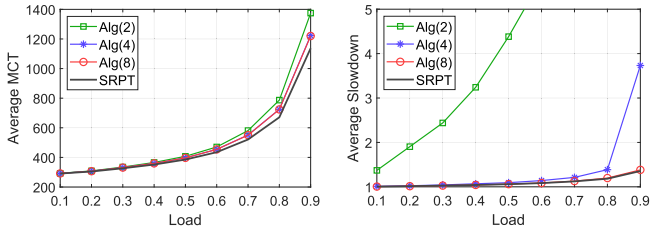
- D/G/1: we suppose that the inter-arrival time of messages is deterministic and equals to $\frac{1}{\lambda}$.
- G/G/1: we suppose that the inter-arrival time of messages follows the uniform distribution on the support $[0, \frac{2}{\lambda}]$.

In the above two scenarios, the message arrival rates are both λ. We will calculate the optimal ADS policy $\boldsymbol{x}^\star$ based on



(a) MCT (b) Slowdown

Fig. 9. Deterministic inter-arrival time.



(a) MCT (b) Slowdown

Fig. 10. Uniformly distributed inter-arrival time.

the M/G/1 model under the arrival rate λ, and adopt the derived ADS policy to the above two scenarios. In each simulation run, we randomly generate 10,000 messages and simulate the scheduling policies $\texttt{Alg}(K)$ and SRPT.

Fig. 9 corresponds to the D/G/1 scenario. The horizontal axis in each sub-figure represents the load λ/μ. The vertical axises in the two sub-figures represent the average MCT and slowdown, respectively. Note that the two-priority ADS policy $\texttt{Alg}(2)$ can already maintain the performance of SRPT for the light-load regime (e.g., $\lambda/\mu < 0.4$). In the heavy-load regime (i.e., $\lambda/\mu > 0.4$), both the ADS policies $\texttt{Alg}(4)$ and $\texttt{Alg}(8)$ perform almost the same as SRPT.

Fig. 10 corresponds to the G/G/1 scenario, where the inter-arrival time is uniformly distributed. Comparing Fig. 10(b) to Fig. 9(b) shows that the randomness of the arrival process results in the performance degradation for $\texttt{Alg}(2)$ and $\texttt{Alg}(4)$. Nevertheless, the circle curve in Fig. 10(b) shows that the ADS policy $\texttt{Alg}(8)$ still retains the performance of SRPT.

VI. Packet-Level Experiment

We carry out extensive packet-level experiments on NS-3 [23] to evaluate the performance of the optimal ADS design. Specifically, we consider the network topology shown in Fig. 11(a), where there are 62 senders, a receiver, and a switch. The bandwidth of each link is 100Gbps. We generate 3000 messages for each sender according to the Poisson process. The message size is drawn from the realistic heavy-tail distribution, where 50% of the messages are of 1KB, 35% are between (1KB,201KB], and 15% are between (201KB,3000KB]. We run the experiments for different scheduling policies under different network load $\{0.3, 0.5, 0.7, 0.9\}$.

A. Average Performance

Fig. 11 shows the average MCT and average slowdown achieved by SRPT and $\texttt{Alg}(K)$ with $K \in \{2, 4, 8\}$ in the packet-level experiments. We have two-fold observations:

- As shown in Fig. 11(b), the average MCT achieved by these scheduling policies is comparable. As we mentioned, this is because that a few long messages dominate the average MCT. Hence the average MCT is not that informative when there are many short messages.
- As shown in Fig. 11(c), the average slowdown achieved by $\texttt{Alg}(2)$ still has a significant gap compared to SRPT. The average slowdown of $\texttt{Alg}(4)$ and SRPT is comparable for the light-load regime (i.e., 0.3, 0.5, and 0.7). Moreover, $\texttt{Alg}(8)$ achieves a similar average slowdown compared to SRPT even for the heavy load 0.9.

The above observations indicate that the optimal ADS policy $\texttt{Alg}(8)$ achieves the similar average MCT and slowdown compared to SRPT in the packet-level experiments.

B. Performance of Emulating SRPT

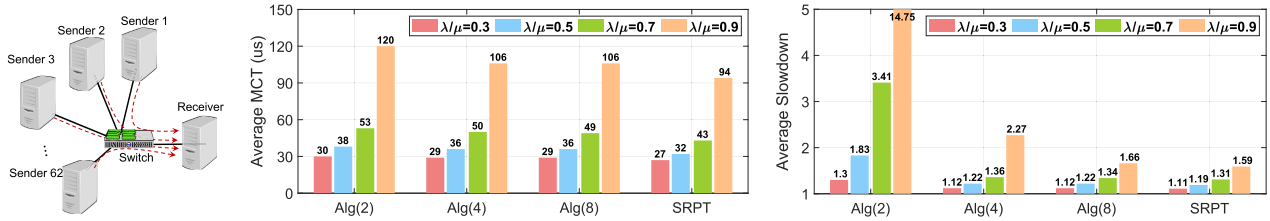
We demonstrate how well $\texttt{Alg}(K)$ performs in terms of emulating SRPT among the preemptive size-based policies (e.g., PSJF and $\texttt{ES}(K)$). Fig. 12 plots the pack-level experiment results achieved by PSJF, $\texttt{ES}(K)$, and $\texttt{Alg}(K)$ at load 0.9. The two sub-figures show the overall average slowdown and percentile slowdown, respectively.

Fig. 12(a) plots the average slowdown across all messages. It leads to two observations that are slightly different from the flow-level simulation results in Section V.

- In Fig. 12(a), PSJF is a little better than $\texttt{Alg}(8)$ in terms of the average slowdown. In Fig. 4(b), however, $\texttt{Alg}(8)$ is a little better than PSJF.
- In Fig. 12(a), the average slowdown gap between $\texttt{ES}(K)$ and $\texttt{Alg}(K)$ in the packet-level experiments is smaller than that in the flow-level simulation shown in Fig. 6(b).

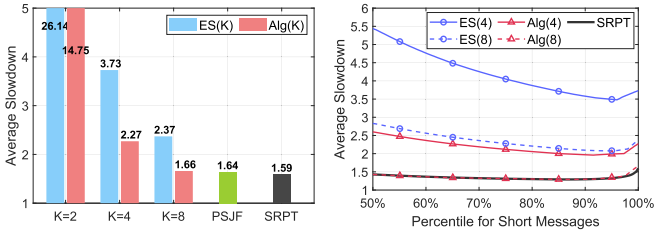
The above inconsistency is because that the flow-level simulation omits the packet-level dynamics, e.g., the packets of a message sequentially arrive at the switch. Nevertheless, the optimal ADS policy $\texttt{Alg}(K)$ still has a significant advantage over PSJF and $\texttt{ES}(K)$. First, it requires a huge number of FIFO queues to implement PSJF on the switch, while $\texttt{Alg}(8)$ needs only eight FIFO queues and achieves the comparable performance to PSJF. Second, Fig. 12(a) only shows the overall average slowdown, which still overlooks some critical information especially for short messages. This motivates us to compare $\texttt{ES}(K)$ and $\texttt{Alg}(K)$ based on the percentile slowdown in the following.

Fig. 12(b) plots the average slowdown across different percentiles for short messages. Based on the adopted message size distribution, for example, when the horizontal axis is 85%, the corresponding vertical axis represents the average slowdown across the messages of sizes {1KB,2KB,…,201KB}. In Fig. 12(b), the black curve without marker corresponds to SRPT. The two circle curves represent cases $\texttt{ES}(4)$ and $\texttt{ES}(8)$. The two triangle curves represent cases $\texttt{Alg}(4)$ and $\texttt{Alg}(8)$. We find that the solid triangle curve (i.e., $\texttt{Alg}(4)$) is much



(a) Topology (b) Average MCT (c) Average slowdown

Fig. 11. Packet-level experiment topology and results.



(a) Overall average slowdown (b) Average over different percentiles

Fig. 12. Comparing PSJF, ES(K) and Alg(K) in the packet-level experiments under the load 0.9.

closer to the black curve (i.e., SRPT) than the solid circle curve (i.e., ES(4)) at any percentile. This means that Alg(4) is better than ES(4) in terms of emulating SRPT. Moreover, we find that the dash triangle curve (i.e., Alg(8)) almost overlaps with the black curve (i.e., SRPT) at any percentile. This shows that the ADS policy Alg(8) is capable of emulating the true SRPT, especially for the short messages.

VII. Conclusion

This paper studies approximate and deployable SRPT (ADS) design. The ADS design aims to emulate SRPT relying only on a few FIFO queues and the original message size information. We first characterize a wide range of ADS policies via a unified framework as a binary vector, and then derive the steady-state MCT, slowdown, and impoliteness in the M/G/1 setting. We formulate the optimal ADS policy design as a non-linear combinatorial optimization problem, aiming to minimize the average steady-state MCT given the available FIFO queues in the switch. We also take into account the proportional fairness and temporal fairness issue based on the maximal slowdown and impoliteness. The optimal ADS design problem is NP-hard, and does not exhibit monotonicity and sub-modularity. We leverage the decomposable structure of the problem, and devise an efficient algorithm to solve the optimal ADS policy. We also carry out extensive flow-level simulations and packet-level experiments based on the real-world heavy-tail message size distributions. The results show that the optimal ADS policy installed on eight FIFO queues is capable of emulating the true SRPT. Furthermore, this paper focuses on the discrete and finite message sizes. The optimal ADS design for the continuous case still remains as an open problem, and needs more investigations in the future.

Appendix

Proof of Proposition 1: The proof of this proposition consists of two parts. First, we show that the messages of sizes in the subset $\{l_n(\boldsymbol{x})+1, l_n(\boldsymbol{x})+2, \ldots, r_n(\boldsymbol{x})\}$ have the same priority (thus will be assigned to the same FIFO queue). According to the definitions in (4) and (5), we have

$$x_j = \begin{cases} 0, & \text{if } l_n(\boldsymbol{x})+1 \le j \le r_n(\boldsymbol{x})-1, \\ 1, & \text{if } j = r_n(\boldsymbol{x}). \end{cases} \tag{25}$$

Based on the definition of the N-dimensional vector $\boldsymbol{x} \in \mathcal{X}$, (25) indicates that the messages of sizes in $\{l_n(\boldsymbol{x})+1, l_n(\boldsymbol{x})+2, \ldots, r_n(\boldsymbol{x})\}$ have the same priority. Second, the priority level of the message subset $\{l_n(\boldsymbol{x})+1, l_n(\boldsymbol{x})+2, \ldots, r_n(\boldsymbol{x})\}$ depends on the number of the preemption points among the set $\{1, 2, \ldots, r_n(\boldsymbol{x})\}$. Hence we have $K_n(\boldsymbol{x}) = \sum_{i=1}^{r_n(\boldsymbol{x})} x_i$. This completes the proof. □

Proof of Proposition 2: The proof of this proposition consists of two steps. First, the size-i messages' steady-state MCT $T_i(\boldsymbol{x})$, slowdown $S_i(\boldsymbol{x})$, and impoliteness $R_i(\boldsymbol{x})$ are

$$\begin{aligned} T_i(\boldsymbol{x}) &= \left[\frac{V(r_i(\boldsymbol{x}))}{1-\rho(r_i(\boldsymbol{x}))} + i\right] \frac{1}{1-\rho(l_i(\boldsymbol{x}))}, \\ S_i(\boldsymbol{x}) &= \left[\frac{V(r_i(\boldsymbol{x}))}{1-\rho(r_i(\boldsymbol{x}))} \cdot \frac{1}{i} + 1\right] \frac{1}{1-\rho(l_i(\boldsymbol{x}))}, \\ R_i(\boldsymbol{x}) &= \rho(l_i(\boldsymbol{x})), \end{aligned} \tag{26}$$

which depend on $l_i(\boldsymbol{x})$ and $r_i(\boldsymbol{x})$.

Second, the policy $\boldsymbol{x} \in \mathcal{X}$ satisfying $x_n = 1$ indicates that we have $l_i(\boldsymbol{x}) < r_i(\boldsymbol{x}) \le n$ for any $i \le n$.

The two aspects above show that (26) does not depend on $x_j \in \{0, 1\}$ for any $j > n$, which completes the proof. □

Acknowledgment

The authors would like to thank the anonymous reviewers and the editor for their useful comments that have helped improve this article.

References

[1] Z. Wang, J. Ye, D. Lin, Y. Chen, and J. C. S. Lui, "Designing approximate and deployable SRPT scheduler: A unified framework," in *Proc. IEEE/ACM IWQoS*, Jun. 2021, pp. 1–6.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, 2008.

[3] M. Noormohammadpour and C. S. Raghavendra, "Datacenter traffic control: Understanding techniques and tradeoffs," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1492–1525, 2nd Quart., 2017.

[4] D. R. Smith, "Technical note—A new proof of the optimality of the shortest remaining processing time discipline," *Oper. Res.*, vol. 26, no. 1, pp. 197–199, Feb. 1978.
[5] *Cisco Nexus 9000 Series NX-OS Quality of Service Configuration Guide*, Dynamic Packet Prioritization, San Jose, CA, USA, 2021.
[6] M. Alizadeh *et al.*, "PFAbric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 435–446, 2013.
[7] W. Bai, L. Chen, K. Chen, D. Han, C. Tian, and H. Wang, "Information-agnostic flow scheduling for commodity data centers," in *Proc. USENIX NSDI*, 2015, pp. 455–468.
[8] P. X. Gao, A. Narayan, G. Kumar, R. Agarwal, S. Ratnasamy, and S. Shenker, "PHost: Distributed near-optimal datacenter transport over commodity network fabric," in *Proc. CoNEXT*, 2015, pp. 1–12.
[9] B. Montazeri, Y. Li, M. Alizadeh, and J. Ousterhout, "HOMA: A receiver-driven low-latency transport protocol using network priorities," in *Proc. SIGCOMM*, 2018, pp. 221–235.
[10] Y. Lu *et al.*, "One more queue is enough: Minimizing flow completion time with explicit priority notification," in *Proc. INFOCOM*, 2017, pp. 1–9.
[11] A. Mushtaq, R. Mittal, J. McCauley, M. Alizadeh, S. Ratnasamy, and S. Shenker, "Datacenter congestion control: Identifying what is essential and making it practical," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 3, pp. 32–38, 2019.
[12] M. Alizadeh *et al.*, "Data center TCP (DCTCP)," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 63–74, Aug. 2010.
[13] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 127–138, 2012.
[14] P. MacArthur and R. D. Russell, "A performance study to guide RDMA programming decisions," in *Proc. HPCC*, 2012, pp. 778–785.
[15] H. Takagi, "Queuing analysis of polling models," *ACM Comput. Surv.*, vol. 20, no. 1, pp. 5–28, Mar. 1988.
[16] T. O'Donovan, "Direct solutions of M/G/1 priority queueing models," *RAIRO-Oper. Res.*, vol. 10, no. 1, pp. 107–111, 1976.
[17] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 1, pp. 56–62, 2006.
[18] N. Bansal and M. Harchol-Balter, "Analysis of SRPT scheduling: Investigating unfairness," in *Proc. SIGMETRICS*, 2001, pp. 279–290.
[19] A. Wierman and M. Harchol-Balter, "Classifying scheduling policies with respect to unfairness in an M/GI/1," in *Proc. SIGMETRICS*, 2003, pp. 238–249.
[20] A. Wierman, "Scheduling for today's computer systems: Bridging theory and practice," Ph.D. dissertation, School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2007.
[21] A. Wierman, "Fairness and scheduling in single server queues," *Surveys Operations Res. Manage. Sci.*, vol. 16, no. 1, pp. 39–48, Jan. 2011.
[22] L. Kleinrock, "A conservation law for a wide class of queueing disciplines," *Nav. Res. Logistics*, vol. 12, no. 2, pp. 181–192, 1965.
[23] *The Network Simulator NS-3*. Accessed: Mar. 2021. [Online]. Available: https://www.nsnam.org
[24] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. SIGCOMM*, 2010, pp. 267–280.
[25] M. Harchol-Balter, K. Sigman, and A. Wierman, "Asymptotic convergence of scheduling policies with respect to slowdown," *Perform. Eval.*, vol. 49, nos. 1–4, pp. 241–256, 2002.
[26] R. Mittal *et al.*, "Revisiting network support for RDMA," in *Proc. SIGCOMM*, 2018, pp. 313–326.

Zhiyuan Wang received the B.Eng. degree in information engineering from Southeast University, Nanjing, in 2016, and the Ph.D. degree from the Department of Information Engineering, The Chinese University of Hong Kong, in 2019. From 2019 to 2021, he was a Post-Doctoral Fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. He is currently an Associate Professor with the School of Computer Science and Engineering, Beihang University. His research interests include network science, game theory, mean field theory, and online learning.

Jiancheng Ye (Member, IEEE) received the B.E. degree in network engineering from Sun Yat-sen University, China, in 2008, the M.Phil. degree in computer science and engineering from The Hong Kong University of Science and Technology in 2011, and the Ph.D. degree in computer networking from The University of Hong Kong in 2018. From 2011 to 2014, he was a Software Engineer with Harmonic Inc. He was a Post-Doctoral Fellow with The University of Hong Kong from 2018 to 2019. He is currently a Researcher with Network Technology Laboratory, Huawei Technologies, Hong Kong. His research interests include congestion control, queue management, optimization of computer networks, edge computing, and online learning. He is a member of ACM.

Dong Lin received the bachelor's degree in computer science from the Beijing University of Aeronautics and Astronautics in 2005, the master's degree in computer science from Tsinghua University in 2008, and the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology in 2012. He was a member of ClusterTech from 2012 to 2014, managed to develop cloud management platforms for both public and private cloud infrastructures. He was a Research Associate Professor at SUSTech in 2018 and 2019. He joined Huawei as a Researcher in 2014. His research interests include data center networks, information-centric networks, content delivery networks, and complex networks.

Yipei Chen received the B.S. degree in mathematics and applied mathematics from Sun Yat-sen University, Guangzhou, China, in 2017, and the Ph.D. degree in mathematics from The Hong Kong University of Science and Technology, Hong Kong, China, in 2021. From December 2020 to June 2021, he was a Research Intern at Huawei Hong Kong Research Center, focusing on flow scheduling and congestion control.

John C. S. Lui (Fellow, IEEE) received the Ph.D. degree in computer science from the University of California at Los Angeles. He is currently the Choh-Ming Li Chair Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His current research interests include machine learning, online learning, such as multi-armed bandit and reinforcement learning, network science, future internet architectures and protocols, network economics, network/system security, and large scale storage systems. He is an Elected Member of the IFIP WG 7.3, a fellow of ACM, and a Senior Research Fellow of the Croucher Foundation. He received various departmental teaching awards and the CUHK Vice Chancellor's Exemplary Teaching Award. He was a co-recipient of the Best Paper Award in the IFIP WG 7.3 Performance 2005, IEEE/IFIP NOMS 2006, SIMPLEX 2013, and ACM RecSys 2017. He was the Past Chair of the ACM SIGMETRICS (2011–2015).