Bilevel Optimization for Provisioning Heterogeneous Traffic in Deterministic Networks

Jing Zhu, Dan Wang, Jiao Xing, Shuxin Qin, Gaofeng Tao, Pingping Chen, Zuqing Zhu, *Fellow, IEEE*

***Abstract*—Due to the capabilities of providing extremely low packet loss and bounded end-to-end latency, deterministic networking (DetNet) has been considered as a promising technology for emerging time-sensitive applications (e.g., industrial control and smart grids) in IP networks. To provide deterministic services, the operator needs to address the routing and scheduling problem. In this work, we study the problem from a novel prospective, i.e., the problem should be optimized not only for deterministic traffic, but also for normal traffic to coexist with the former. Specifically, we redefine the problem as bandwidth allocation, routing and scheduling (BaRS), and model this problem as a bilevel optimization which consists of an upper-level optimization and a lower-level optimization. The upper-level optimization allocates link bandwidth between deterministic and normal traffic to maximize the available bandwidth for normal traffic on the premise of accepting a certain portion of deterministic bandwidth; the lower-level optimization determines specific routing and scheduling solutions for deterministic traffic to maximize the number of accepted deterministic flows. We first formulate the bilevel optimization as a bilevel mixed integer linear programming (BMILP). Then, we propose an exact algorithm based on cutting planes to solve it exactly, and propose an approximation algorithm based on two-level relaxations and randomized rounding to solve it effectively and time-efficiently. Extensive simulations are conducted and the results verify the effectiveness of our proposals in balancing the tradeoff between the available bandwidth for normal traffic and the number of accepted deterministic flows.**

***Index Terms*—Deterministic networking, Normal and deterministic traffic, Bandwidth allocation, routing and scheduling, Bilevel optimization.**

I. Introduction

In the past years, the prosperity of 5G and Internet protocols has stimulated industrial applications transitioning from dedicated vertical solutions to general IP-based infrastructures [1]. These applications, such as smart grids and industrial control, require IP networks to provide deterministic services [2]. On the other hand, traditional IP networks cannot provide rigorous quality of service (QoS) guarantees, although certain services can be handled in higher priority acquiring improved QoS. Under the circumstances, deterministic networking (DetNet) has attracted intensive attentions for its capabilities of providing extremely low packet loss, bounded end-to-end latency and jitter, which can be achieved by employing various mechanisms, such as explicit routing, queuing and scheduling [3]. In particular for bounded end-to-end latency, an explicit path ensures that the latency from links between the two ends is constant; a well-designed queuing and scheduling mechanism can provision sufficient buffer at nodes along the path to avoid packet loss, and meanwhile control the latency from nodes to be bounded.

The IETF DetNet working group has drafted several queuing mechanisms to enable bounded end-to-end latency, among which cyclic specified queuing and forwarding (CSQF) based on Segment routing (SR) is progressing through commercialization and thus demonstrates greater maturity than the others [4]–[8]. Specifically, CSQF requires a frequency-synchronized network and logically divides the transmission at ports into equal cycles. It applies several queues (more than 2) on each egress port to open and close in a round-robin pattern. During any cycle, one queue is for transmission while the others are for reception. Figs. 1(a)-(b) illustrate an example of CSQF, where 3 queues are used for deterministic flows and work cyclicly. Given a deterministic flow, SR employs SIDs in the header of packets to explicitly specify to which egress port and in which cycle the packets should be transmitted from the source. Fig. 1(c) shows an example of an SID inside a packet. Assuming that the packet arrives in cycle 1 at the second hop, the SID indicates that it should be forwarded to port 2 and enter queue 3 according to the mapping between transmission cycles and queues.

Although possessing CSQF and SR, a network controller still needs to carefully determine the egress ports (i.e., routing) and transmission cycles (i.e., scheduling) to provision deterministic flows. Fig. 1(d) shows an example of the routing and scheduling (RS) for a deterministic flow. Considering the flow from node A to node E, the route is selected as node A $\rightarrow$ node C $\rightarrow$ node D $\rightarrow$ node E. The packets are sent from node A during cycle 1. When arriving at node C during cycle 2, they are scheduled to be transmitted in the next cycle. Then at node D, they are scheduled to be transmitted two cycles later, i.e., in cycle 6. Previously, the problem of RS has been studied in [9]–[13]. Nevertheless, these works only think over deterministic traffic to maximize its bandwidth or the number of accepted flows as much as possible. Note that, in addition to deterministic traffic, there are plenty of applications generating normal traffic, which accounts for a substantial part of network traffic. To support such traffic, it is essential for DetNet to coexist well with existing QoS mechanisms [3]. This means that: 1) when deterministic traffic are scheduled with specific transmission opportunities, normal traffic should also be provided with sufficient transmission opportunities to avoid starvation; 2) on the premise of not disrupting deterministic

J. Zhu, D. Wang, J. Xing, S. Qin, G. Tao and P. Chen are with Purple Mountain Laboratories, Nanjing, Jiangsu 211111, P. R. China.

Z. Zhu is with the School of Information Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, P. R. China (email: zqzhu@ieee.org).

Manuscript received on January 2, 2025, revised on April 18, 2025.

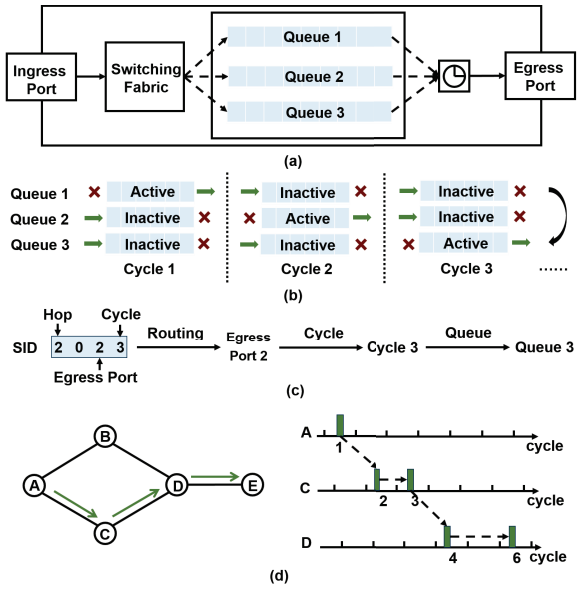

Fig. 1. Illustrations: (a) simplified switching device supporting CSQF, (b) scheduling principle of CSQF, (c) SID of SR for CSQF, (d) example of RS.

traffic, transmission opportunities dedicated to but not utilized by deterministic traffic should be available to normal traffic. This complicates the problem of RS and motivates us to revisit it.

In this work, we incorporate the problem of bandwidth allocation between deterministic and normal traffic into consideration, and redefine the problem as bandwidth allocation, routing and scheduling (BaRS) to maximize the number of accepted deterministic flows and maximize the bandwidth available for normal traffic simultaneously. Considering that the two maximizations contradict to each other and can hardly be integrated into a single-level optimization, we model the BaRS problem as a bilevel optimization, consisting of an upper-level and a lower-level optimizations. Specifically, the upper-level optimization allocates link bandwidth between deterministic and normal traffic such that the bandwidth available for normal traffic is maximized, while the lower-level one determines RS for deterministic traffic to achieve a maximized number of accepted flows. Furthermore, the two levels correlate with each other in that the upper-level optimization constrains the RS for deterministic traffic in the lower-level one while the lower-level optimization evaluates the upper-level one. We first formulate the bilevel optimization as a bilevel mixed integer linear programming (BMILP). Then, we propose an exact algorithm based on cutting planes and an approximation algorithm based on two-level relaxations and randomized rounding. Finally, extensive simulations verify the effectiveness of our proposals.

The rest of the paper is organized as follows. Section II elaborates related works. Section III proposes the BaRS problem and describes the system model. Section IV formulates the bilevel optimization and analyzes its complexity. The exact and approximation algorithms are described in Section V and Section VI respectively. Simulations are provided in Section VII, and conclusions are drawn in Section VIII.

II. Related Work

In past decades, QoS has always been one of the essential issues in networks. To achieve satisfying QoS, researchers have proposed various mechanisms in scheduling, which we could simply categorize into normal ones (e.g., [14]–[17]) and deterministic ones (e.g. [4], [18]–[21]). Normal mechanisms applied to normal traffic, which provided differentiated but still statistical QoS. For instance, weighted fair queueing (WFQ) [17] scheduled queues in a round-robin pattern, and each time allowed one queue to transmit a certain volume of bits according to the queue's weight. To achieve strict QoS guarantees, deterministic mechanisms were proposed. For example, for layer-2 networks, IEEE 802.1Qch proposed cycle queuing and forwarding (CQF) [20], where 2 queues open and closed alternately in cycles to transmit and receive packets. By employing more queues for slack synchronization and advanced scheduling, CQF are extended for IP networks to achieve improved flexibility and scalability, typically CSQF. When deploying CSQF, the scheduling is more flexible and therefore the RS problem becomes more intractable.

To facilitate implementations of deterministic mechanisms, various works have studied the RS problem in [9]–[13], [22]–[28]. In layer-2 networks, the researchers of [22]–[28] based on CQF or its variants, and investigated the RS problem with methods such as divisibility theory and deep reinforcement learning (DRL). Towards IP networks, [10] proposed advanced CQF that adds 2 backup queues and design a heuristic algorithm for the RS problem to maximize the number of accepted deterministic flows. However, due to the working principle of CQF, the scheduling problem therein merely needed to determine the cycle at the network border where packets are injected. When deploying CSQF, the scheduling is more flexible and therefore the RS problem becomes more intractable. In [9], Krolikowski et al. studied the problem and proposed two algorithms to maximize the accepted bandwidth of deterministic traffic. Despite mentioning that a percentage (e.g., 50%) of bandwidth should be allocated for normal traffic, they did not delve into how to allocate the bandwidth. In [11], they studied 1+1 protection for deterministic traffic and introduced spacing constraints for the RS problem. To solve the problem, two heuristic algorithms (i.e., the greedy and the Tabu-search) were proposed to maximize the number of accepted deterministic flows. The researchers of [12] considered that the flow's rate could be adjusted at the network border, and redefined the routing problem with multi-variable constraints. A Lagrange relaxation based heuristic was proposed to solve the routing problem. In [13], Yu et al. classified deterministic flows into hard ones and soft ones, and designed a DRL based scheduler to solve the RS problem with the objective of maximizing the number of hard flows and the utilities of soft flows. As stated above, existing studies did not consider that deterministic and normal traffic coexist in IP networks, and to the best of our knowledge, the BaRS problem has not been investigated. In this work, we will study the BaRS problem with a fancy optimization, i.e., bilevel optimization, which has been successfully applied to computer networks [29]–[31].

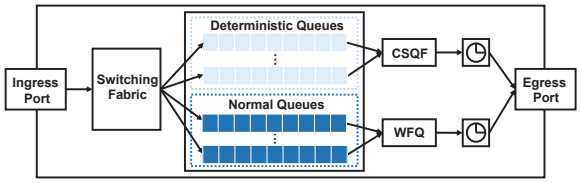

Fig. 2. A simplified switching device supporting deterministic and normal traffic.

III. Problem Description

A. BaRS Problem

Fig. 2 depicts a simplified switching device supporting transmitting deterministic and normal traffic in DetNet. As it shows, each egress port is associated with several logical queues (typically 8), where some (more than 2) are dedicated for deterministic traffic while the others are for normal traffic. Deterministic queues are scheduled with CSQF, while the normal ones can be scheduled with normal mechanisms mentioned in section II, e.g., WFQ. The time of each port is divided into equal cycles (e.g., 10 us) and packet transmission operates in cycles. Each cycle, both a deterministic and a normal queue can access the bandwidth such that the deterministic one is drained and the normal one discharges a certain amount of traffic. Here, the problem of bandwidth allocation between deterministic and normal traffic arises. Qualitatively speaking, no matter for deterministic or normal traffic, the more bandwidth is allocated, higher throughput or better QoS tends to be achieved. Allocating more bandwidth for either one would incur traffic losses or service degradation for the other one. Hence, the problem should be taken seriously.

In this work, we integrate the bandwidth allocation problem with the RS problem to study the BaRS problem, which is inherently a hierarchical decision process. Concretely, the controller first allocates the bandwidth of each port to maximize the bandwidth available for normal traffic on the premise of certain accepted deterministic traffic, and then addresses the RS problem for deterministic traffic to maximize the number of accepted flows. Note that, the bandwidth allocation and the RS interact, and the two maximizations contradict to each other. Namely, bandwidth allocation constrains the bandwidth for deterministic traffic in RS, while RS conversely evaluates the bandwidth available for normal traffic. To capture such characteristics of the BaRS problem, a single-level optimization is no longer applicable, which motivates us to adopt a brand-new optimization, i.e., the bilevel optimization in Section IV.

B. System Model

DetNet is modelled as a directed graph $G(V, E)$, where nodes V represent DetNet-enabled swithes/routers and are connected with directed links E. According to CSQF, DetNet is required to be frequency-synchronized, and time is divided into equal cycles c across the network [4]. Without loss of generality, the cycles can be assumed starting at the same time [9]. Blocks of consecutive cycles consist of hypercycles H, in each of which the network behavior repeats. For each node $v \in V$, the number of queues dedicated for deterministic traffic per egress port is N. Each link $e_{u,v} \in E$ has the bandwidth as $b_{u,v}$ and the length as $l_{u,v}$. The possible portions of bandwidth that can be allocated for deterministic traffic are denoted as set B.

In diverse industries, the patterns of traffic demanding deterministic forwarding can be time-triggered (i.e., periodic) or event-triggered (i.e., sporadic) [32]. Since CSQF is proposed for time-triggered traffic, we will only consider this type of traffic in this work, and model deterministic traffic as a set of time-triggered flows F. Each flow $f \in F$ can be defined as a five-tuple $(\ell_f, \tau_f, \omega_f, Q_f, P_f)$. Specifically, ℓ_f is the packet size, τ_f is the maximum end-to-end delay that the flow can tolerate, ω_f is the period in cycles. Q_f is the pattern of packet arrivals, and $Q_{f,i}$ denotes the number of packets arrived in the i-th cycle. P_f is the precomputed paths for the flow. To provision flow f, the controller needs to determine the RS, i.e., selecting a feasible path and assigning a transmission cycle at each node along the path except the destination. At each node, the transmission can be delayed by a few cycles for successful scheduling, where the number of cycles should be no more than $N-2$ according to CSQF. Moreover, the RS should satisfy the maximum end-to-end delay and bandwidth constraints to get accepted. Note that, we consider that the propagation delay and the queueing delay constitute the end-to-end delay[1]. In this work, we set the hypercycle H as the least common multiple of flow periods, and hence study the BaRS problem within a hypercycle.

IV. Bilevel Optimization

We model the BaRS problem as a bilevel optimization, which is a branch of mathematical optimization that some variables are constrained to be the solution of another optimization [33]. Specifically, the upper-level optimization addresses bandwidth allocation between normal and deterministic traffic to maximize the available bandwidth for normal traffic on the premise of certain accepted deterministic traffic; the lower-level optimization addresses RS for deterministic flows to maximize the number of accepted flows.

A. BMILP Formulation

We formulate the bilevel optimization as a BMILP. Table I lists the parameters of the BMILP.

1) Upper-level Optimization: It is to allocate the bandwidth on each link between deterministic and normal traffic. Table II summarizes the variables.

Objective:

$$\text{Maximize} \quad \Omega_n = \Omega_n^1 + \lambda \cdot \Omega_n^2 \tag{1}$$

The upper-level optimization maximizes the bandwidth available for normal traffic, which consists of two parts Ω_n^1 and Ω_n^2. Ω_n^1 denotes the part that is allocated for normal traffic, while Ω_n^2 denotes the part that is allocated for but not fully

[1]We should point out that there are also the processing delay and the transmission delay constituting the end-to-end delay. Here, the transmission delay, different from the propagation delay, is the time taken to push all the bits of a packet onto the transmission medium. Since they are almost independent of RS, we treat them as constants and do not consider them in this work.

TABLE I
PARAMETERS OF THE BMILP

$G(V, E)$	the topology of the DetNet
$b_{u,v}$	the bandwidth of link $e_{u,v} \in E$
$l_{u,v}$	the propagation delay in the number of cycles of link $e_{u,v} \in E$
N	the number of queues for deterministic traffic
c	the cycle in seconds
H	the hypercycle in the number of cycles
F	the set of deterministic flows
ℓ_f	the packet size of flow $f \in F$
τ_f	the maximum tolerable end-to-end delay in the number of cycles of flow $f \in F$
Q_f	the pattern of packet arrivals of flow f
P_f	the set of candidate paths for flow $f \in F$
$\|p\|$	the number of links in the path $p \in P_f$
B	the set of possible portions of bandwidth that can be allocated for deterministic traffic
ρ	the ratio of accepted deterministic traffic
λ	a weight coefficient
M	a large positive for linearization

utilized by deterministic traffic. λ is a coefficient to weigh the importance of them.

TABLE II
UPPER-LEVEL VARIABLES

$x_{u,v,i}$	the boolean variable that equals 1 if the i-th ratio in B is selected for deterministic traffic on link $e_{u,v} \in E$, and 0 otherwise
$\bar{b}_{u,v}$	the variable that denotes the bandwidth allocated for deterministic traffic on link $e_{u,v} \in E$
$y_{f,p}$	the boolean variable that equals 1 if path $p \in P_f$ is selected for flow $f \in F$, and 0 otherwise
Ω_n	the variable that denotes the total bandwidth available for normal traffic
Ω_n^1	the variable that calculates the bandwidth allocated for normal traffic
Ω_n^2	the variable that calculates the bandwidth allocated for but not utilized by deterministic traffic
Ω_d	the variable that calculates the number of accepted deterministic flows

Constraints:

$$\sum_{i=0}^{|B|-1} x_{u,v,i} = 1, \quad e_{u,v} \in E. \tag{2}$$

$$\bar{b}_{u,v} = \sum_{i=0}^{|B|-1} (x_{u,v,i} \cdot B_i) \cdot b_{u,v}, \quad e_{u,v} \in E. \tag{3}$$

Eqs. (2)-(3) specify the bandwidth that is allocated for deterministic flows on each link.

$$\Omega_d \geq \rho \cdot \sum_{f \in F} \sum_{i=0}^{H-1} \frac{\ell_f \cdot Q_{f,i}}{H \cdot c} \tag{4}$$

Eq. (4) ensures a minimum ratio of accepted deterministic traffic.

$$\Omega_n^1 = \sum_{e_{u,v} \in E} (b_{u,v} - \bar{b}_{u,v}) \tag{5}$$

$$\Omega_n^2 = \sum_{e_{u,v} \in E} \bar{b}_{u,v} - \sum_{f \in F} \sum_{p \in P_f} \sum_{i=0}^{H-1} \frac{\ell_f \cdot Q_{f,i}}{H \cdot c} \cdot |p| \cdot y_{f,p} \tag{6}$$

$$\Omega_d = \sum_{f \in F} \sum_{p \in P_f} \sum_{i=0}^{H-1} \frac{\ell_f \cdot Q_{f,i}}{H \cdot c} \cdot y_{f,p} \tag{7}$$

Eqs. (5)-(7) calculate Ω_n^1, Ω_n^2 and Ω_d respectively.

TABLE III
LOWER-LEVEL VARIABLES

$y_{f,p}$	the boolean variable that equals 1 if flow $f \in F$ is routed on path $p \in P_f$, and 0 otherwise
$o_{f,p,i}$	the integer variable that denotes the delay in cycles on the i-th link of path $p \in P_f$ for flow $f \in F$
$q_{f,p,i}$	the total queuing delay in cycles that flow $f \in F$ has experienced on the i-th link of path $p \in P_f$
$r_{f,p,i}$	the total propagation delay in cycles that flow $f \in F$ has experienced on the i-th link of path $p \in P_f$
$w_{f,p,i}^j$	the boolean variable that equals 1 if the accumulated cycle shift for flow $f \in F$ on the i-th link of path $p \in P_f$ is j, and 0 otherwise
$m_{f,p,i}^j$	the auxiliary integer variable that calculates the number of packets for flow $f \in F$ in the j-th cycle on the i-th link of path $p \in P_f$

2) Lower-level Optimization: It is to determine RS for deterministic traffic. Note that the two optimizations correlate with each other. Specifically, the variables $\{\bar{b}_{u,v}\}$ in the upper-level optimization will serve as parameters in the lower-level one, while the solutions $\{y_{f,p}\}$ of the lower-level optimization will conversely evaluate the objective of the upper-level one. Table III summarizes the variables.

Objective:

$$\text{Maximize} \quad \sum_{f \in F} \sum_{p \in P_f} y_{f,p} \tag{8}$$

The lower-level objective is to maximize the number of accepted deterministic flows.

Constraints:

$$\sum_{p \in P_f} y_{f,p} \leq 1, \quad \forall f \in F. \tag{9}$$

Eq. (9) ensures that each deterministic flow is routed over one path at most.

$$\begin{aligned} o_{f,p,i} + M \cdot (y_{f,p} - 1) \leq N - 2, \\ \forall\ f \in F,\ p \in P_f,\ e_{u,v} = p_i. \end{aligned} \tag{10}$$

Eq. (10) ensures that the scheduling of each flow on each link is within the right range.

$$\begin{cases} q_{f,p,i} = \sum_{i'=0}^{i} o_{f,p,i'}, \\ r_{f,p,i} = \sum_{e_{u,v}=p_0}^{p_i} l_{u,v}, \end{cases} \quad \begin{array}{l} \forall\ f \in F,\ p \in P_f, \\ i \in [0, |p|-1]. \end{array} \tag{11}$$

Eq. (11) calculates the accumulated queuing and propagation delays along the path for each flow.

$$\begin{array}{c} q_{f,p,i} + r_{f,p,i} + M \cdot (y_{f,p} - 1) \leq \tau_f, \\ \forall\ f \in F,\ p \in P_f,\ i = |p| - 1. \end{array} \tag{12}$$

Eq. (12) ensures that the maximum tolerable end-to-end delay of each flow should not be exceeded.

$$\begin{array}{c} y_{f,p} \leq \sum_{j=0}^{H-1} w_{f,p,i}^{j} \leq (1-M) \cdot y_{f,p} + M, \\ \forall\ f \in F,\ i \in [0, |p|-1]. \end{array} \tag{13}$$

$$\begin{array}{c} \sum_{j=0}^{H-1} j \cdot w_{f,p,i}^{j} = (q_{f,p,i} + r_{f,p,i}) \bmod H, \\ \forall\ f \in F,\ i \in [0, |p|-1]. \end{array} \tag{14}$$

Eqs. (13)-(14) figure out whether each flow has packets arrived at a cycle on each link along the path.

$$\begin{array}{c} m_{f,p,i}^{j} = \sum_{k<=j} Q_{f,k} \cdot w_{f,p,i}^{j-k} + \sum_{k>j} Q_{f,k} \cdot w_{f,p,i}^{j-k+H}, \\ \forall\ f \in F,\ i \in [0, |p|-1],\ j \in [0, |H|-1]. \end{array} \tag{15}$$

Eq. (15) precisely computes the number of packets for each flow arrived at each cycle on each link along the path.

$$\begin{array}{c} \frac{1}{c} \cdot \sum_{f \in F} \sum_{p \in P_f} \ell_f \cdot m_{f,p,i}^{j} \leq \bar{b}_{u,v}, \\ \forall\ e_{u,v} \in E,\ j \in [0, H-1],\ p_i = e_{u,v}. \end{array} \tag{16}$$

Eq. (16) ensures that the allocated bandwidth in each cycle on each link should not be exceeded.

B. Complexity Analysis

We can analyze that the formulated bilevel optimization is NP-Hard. To this end, we introduce the high-point relaxation (HPR) of the bilevel optimization, which relaxes the optimality of the lower-level optimization.

Theorem 1. *Given an optimal solution of the HPR, if it is optimal to the lower-level optimization, it is also optimal to the bilevel optimization.*

Proof. The feasible region of the HPR can be expressed as Eq. (17).

$$\mathbb{H} := \{(\text{x, y}) \mid \text{Eqs. (2)-(7), (9)-(16) are satisfied.}\} \tag{17}$$

Here, (x, y) are decision variables in Section IV-A about bandwidth allocation and routing. An optimal solution of the HPR is the one in Eq. (17) that maximizes Eq. (1). Clearly, an optimal solution of the HPR is a feasible solution of the lower-level optimization, and acts as an upper bound on the optimum of the bilevel optimization. On the other hand, the feasible region of the bilevel optimization can be expressed as Eq. (18), where the optimal response of the lower-level optimization is taken into account.

$$\mathbb{B} := \{(\text{x, y}) \mid (\text{x, y}) \in \mathbb{H} \text{ and y maximzes Eq. (8)}\} \tag{18}$$

An optimal solution of the bilevel optimization is the one in Eq. (18) that maximizes Eq. (1). Comparing Eq. (17) and Eq. (18), we prove the theorem. ■

Theorem 1 indicates that given an optimal solution of the HPR, checking whether it is the optimal response of the bilevel optimization is equivalent to checking whether it is the optimal response of the lower-level optimization. We notice that the problem that whether a solution is the optimal response of the lower-level optimization is exactly the decision counterpart of the lower-level optimization, which has been proved NP-Complete in [9]. This means that the problem that given an optimal solution of the HPR, checking whether it is the optimal response of the bilevel optimization is NP-Complete. Therefore, the bilevel optimization is NP-Hard.

V. Exact Algorithm

To the best of our knowledge, mature solvers are yet missing to solve the BMILP. In this section, we propose an algorithm based on cutting planes to solve it exactly.

A. Cutting Plane Design

As stated in Section IV-B, we can write the HPR as Eq. (19) shows, which is a single-level optimization and can be solved with well-known solvers, such as Cplex [34] and Gurobi [35]. Since the objective of the lower-level optimization is ignored by the HPR, a solution that is optimal to the HPR might not be the optimal response of the lower-level optimization, and is bilevel-infeasible according to Eq. (18). Hence, if we hope to get an optimal bilevel solution from the HPR, such bilevel-infeasible solutions should be excluded from the feasible region of the HPR, which can be achieved by adding valid cuts to the HPR.

$$\begin{array}{c} \text{Maximize} \quad \Omega_n \\ \text{s.t.} \quad \text{Eqs. (2)-(7), Eqs. (9)-(16)} \end{array} \tag{19}$$

In the following, we explain how to design such cuts.

Definition 1: Let $\text{z} := \{z_f \mid f \in [0, |F|-1]\}$ be a set of binary variables such that an accepted flow has $z_f = 1$ and a rejected one has $z_f = 0$, and $\mathbb{Z}$ be the set of all possible z.

According to Eq. (9), we have $\sum_{p \in P_f} y_{f,p} = z_f = 1$ for an accepted flow and have $\sum_{p \in P_f} y_{f,p} = z_f = 0$ for a rejected flow. Therefore, any solution of the HPR or the bilevel optimization can be mapped to a specific z. Provided a HPR-optimal but bilevel-infeasible solution, there exists at least a deterministic flow that can be accepted under the bilevel optimization but

is rejected under the HPR. This indicates that any solution that is bilevel optimal and a solution that is HPR-optimal but bilevel-infeasible are mapped to two different z. Assume that a solution that is HPR-optimal but bilevel-infeasible is mapped to z_0. To exclude z_0, a valid cut should ensure that for any $\mathrm{z} \in \mathbb{Z} \backslash \mathrm{z}_0$,

$$\sum_{f \in [0,|F|)} \gamma_f \cdot z_f \leq \gamma_{|F|}, \quad \forall f \in F, \tag{20}$$

and for z_0,

$$\sum_{f \in [0,|F|)} \gamma_f \cdot z_f \geq \gamma_{|F|}, \tag{21}$$

where $\gamma := \{\gamma_f \mid f \in [0,|F|]\}$ are coefficients and $\sum_{f \in [0,|F|)} \gamma_f \neq 0$. To obtain the coefficients γ, we propose the following integer quadratic programming (IQP).

Parameters:

- $\mathbb{Z}$: all possible z.
- z_0: a specific z that a HPR-optimal but bilevel-infeasible solution is mapped to.

Variables:

- γ: the set of coefficients.

Objective:

$$\text{Minimize} \sum_{f \in [0,|F|]} \gamma_f^2 \tag{22}$$

The objective is to minimize the quadratic sum of the coefficients so that the coefficients can be as compact as possible.

Constraints:

$$\sum_{f \in [0,|F|)} \gamma_f \neq 0 \tag{23}$$

$$\sum_{f \in [0,|F|)} \gamma_f \cdot z_f \leq \gamma_{|F|}, \quad \mathrm{z} \in \mathbb{Z} \backslash \mathrm{z}_0 \tag{24}$$

$$\sum_{f \in [0,|F|)} \gamma_f \cdot z_f > \gamma_{|F|}, \quad \mathrm{z} = \mathrm{z}_0 \tag{25}$$

In the IQP, $\mathbb{Z}$ can be obtained through numeration easily, and z_0 is obtained by solving the HPR. As a single-level optimization, the IQP can be solved with Cplex or Gurobi. Hence, we can get γ and design a valid cut for z_0 as Eq. (20) shows.

Fig. 3 illustrates an example of cutting plane design. Assuming 3 flows, all 8 possible z can be represented by the solid points, where each dimension indicates whether a flow is accepted or not. If the red solid point $\mathrm{z}_0 = (1, 1, 0)$ is HPR-optimal but bilevel-infeasible, the corresponding $\{\gamma_f \mid f \in [0, 3]\}$ can be derived with the IQP as $\{1, 1, -1, 1\}$, obtaining a valid cut as $z_0 + z_1 - z_2 = 1$.

B. Algorithm Description

Algorithm 1 describes the whole procedure of the exact algorithm. *Lines* 1-2 are for initialization. The for-loop covering *Lines* 3-7 prepares all possible valid cuts. The while-loop that covers *Lines* 8-21 tries to obtain a bilevel-optimal solution by solving the HPR in iterations. Specifically, *Line* 9 first solves the HPR to obtain $\{\bar{b}_{u,v}\}$, $\{y^h_{f,p}\}$ and $\{o^h_{f,p,i}\}$. With the obtained $\{\bar{b}_{u,v}\}$, *Line* 10 solves the lower-level optimization to obtain $\{y^l_{f,p}\}$ and $\{o^l_{f,p,i}\}$. *Line* 11 calculates the lower-level objectives in Eq. (7) with the obtained $\{y^h_{f,p}\}$ and $\{y^l_{f,p}\}$ as Ω^h_d and Ω^l_d respectively. If $\Omega^h_d = \Omega^l_d$, we obtain a bilevel-optimal solution successfully according to **Theorem 1**, and therefore *Lines* 14-16 return the solution. Otherwise, *Lines* 18-19 select the valid cut that corresponds to $\{y^h_{f,p}\}$ from Ψ, and add it to the HPR to eliminate the current bilevel-infeasible solution.

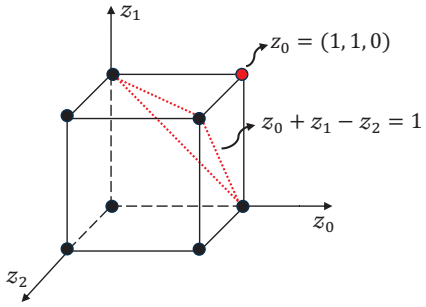

Fig. 3. Example of cutting plane design.

The complexity of *Algorithm* 1 is not polynomial-time. On the one hand, *Lines* 3-7 and *Lines* 9-10 cannot be accomplished in polynomial time. On the other hand, the for-loop covering *Lines* 8-21 can be executed $2^{|F|}$ times at most. In practical, the number of iterations of the for-loop is highly relevant to the parameter ρ in the bilevel optimization, which will be verified with simulations in Section VII.

VI. Approximation Algorithm

Due to the complexity of *Algorithm* 1, it becomes powerless when the problem scale of the bilevel optimization gets larger. In this section, we propose an approximation algorithm.

A. Algorithm Description

Since linear programming relaxation (LPR) and randomized rounding are classical techniques for integer linear programming, the approximation algorithm leverages them to obtain near-optimal solutions whose performance can be analyzed statistically. In this algorithm, we first relax the bilevel optimization to its HPR and then to its LPR as r-HPR. By solving the r-HPR, we get the solutions of bandwidth allocation in real numbers. Next, we proceed to the lower-level optimization and relax it to its LPR as r-LLO. By solving the r-LLO, we get the solutions of RS in real numbers. Then randomized rounding is executed iteratively to obtain integer solutions of the BaRS problem, i.e., $\{\bar{b}_{u,v}\}$, $\{y_{f,p}\}$ and $\{o_{f,p,i}\}$. Note that during randomized rounding, the constraints of the BMILP might be violated, leading to infeasible integer solutions. To avoid this, we introduce a parameter for scaling, i.e., ϵ, with which the probability of generating infeasible solutions can be theoretically analyzed to be bounded [36]. To control the performance of the approximation algorithm, another two parameters are also introduced, i.e., the number of iterations I and the approximation ratio ζ. The overall procedure of the algorithm is provided in *Algorithm* 2 and *Algorithm* 3, where *Algorithm* 3 is a subprocedure of *Algorithm* 2.

Algorithm 1: Exact Algorithm Based on Cutting Plane

input : Parameters in Table I.
output: Results of BaRS as $\{\bar{b}_{u,v}\}$, $\{y_{f,p}\}$ and $\{o_{f,p,i}\}$.

1 initialize $flag$ = 0, $\Psi = \emptyset$;
2 initialize the HPR as Eq. (19);
3 **for** *each* z $\in \mathbb{Z}$ **do**
4 solve the IQP in Eqs. (22)-(25) to obtain γ;
5 construct a valid cut φ with γ as Eq. (20);
6 add φ into Ψ;
7 **end**
8 **while** *flag = 0* **do**
9 solve the HPR to obtain $\{\bar{b}_{u,v}\}$, $\{y^h_{f,p}\}$ and $\{o^h_{f,p,i}\}$;
10 solve the lower-level optimization with $\{\bar{b}_{u,v}\}$ to obtain $\{y^l_{f,p}\}$ and $\{o^l_{f,p,i}\}$;
11 calculate the lower-level objectives with Eq. (7) as Ω^h_d with $\{y^h_{f,p}\}$ and as Ω^l_d with $\{y^l_{f,p}\}$;
12 **if** $\Omega^h_d = \Omega^l_d$ **then**
13 flag = 1;
14 $\{\bar{b}_{u,v} = \bar{b}_{u,v}\}$;
15 $\{y_{f,p} = y^h_{f,p}\}$;
16 $\{o_{f,p,i} = o^h_{f,p,i}\}$;
17 **else**
18 select the φ corresponding to $\{y^h_{f,p}\}$ from Ψ;
19 add φ to the HPR;
20 **end**
21 **end**
22 **return** $\{\bar{b}_{u,v}\}$, $\{y_{f,p}\}$, $\{o_{f,p,i}\}$.

In *Algorithm* 2, *Lines* 1-2 solve the r-HPR and obtain $\{\bar{b}^r_{u,v}\}$ and $\hat{\Omega}_n$ in real numbers. Then with $\{\bar{b}^r_{u,v}\}$, *Lines* 3-4 solve the r-LLO and obtain $\{y^r_{f,p}\}$, $\{o^r_{f,p,i}\}$ and Ω_d in real numbers. *Line* 5 initializes Ω_n. The for-loop covering *Lines* 6-27 describes the procedure of randomized rounding, where each iteration tries to generate a feasible solution of the BMILP and the solution with the best Ω_n is selected over iterations as the desired one. Specifically, the inner for-loop covering *Lines* 7-12 first rounds z_f of each flow to 1 or 0, determining whether the flow can be accepted or not. To avoid infeasible solutions of $\{y_{f,p}\}$, *Lines* 9-11 scale z_f with ϵ before rounding. Although this can only reduce but not definitely eliminate the probability that the constraints in Eq. (4) are violated, an appropriate ϵ could keep such probability in an acceptable level. In case of infeasible solutions, *Lines* 13-14 continue to the next iteration. *Lines* 16-19 guarantee an approximation ratio of the lower-level optimization to its LPR. Then *Line* 20 invokes *Algorithm* 3 and get the results of $\tilde{\Omega}_n$, $\{\tilde{b}_{u,v}\}$, $\{\tilde{y}_{f,p}\}$ and $\{\tilde{o}_{f,p,i}\}$. Finally, *Lines* 21-26 update Ω_n, $\{\bar{b}_{u,v}\}$, $\{y_{f,p}\}$ and $\{o_{f,p,i}\}$ if necessary. In *Algorithm* 3, the for-loop covering *Lines* 1-16 leverages randomized rounding to produce a feasible solution of RS. Specifically, for each flow that are temporarily treated as accepted in *Algorithm* 2, *Lines* 3-7 determine $\{\tilde{y}_{f,p}\}$, and *Lines* 8-12 determine $\{\tilde{o}_{f,p,i}\}$. Otherwise, *Line* 14 updates $\{\tilde{y}_{f,p}\}$ to correctly implementing the following calculations. Then in *Lines* 17-20, a feasible solution of bandwidth allocation is determined as $\{\tilde{b}_{u,v}\}$. Finally, *Line* 21 calculates $\tilde{\Omega}_n$ with Eq. (1).

Algorithm 2: Approximation Algorithm Based on LPR and Randomized Rounding

input : Parameters in Table I and parameters ϵ_1, ϵ_2, I and ζ.
output: BaRS results as $\{\bar{b}_{u,v}\}$, $\{y_{f,p}\}$ and $\{o_{f,p,i}\}$.

1 relax the HPR in Eq. (18) as r-HPR;
2 solve r-HPR to obtain $\{\bar{b}^r_{u,v}\}$ and $\hat{\Omega}_n$ in real numbers;
3 relax the lower-level optimization in Eqs. (7)-(15) as r-LLO;
4 solve r-LLO with $\{\bar{b}^r_{u,v}\}$ to obtain $\{y^r_{f,p}\}$, $\{o^r_{f,p,i}\}$ and Ω_d in real numbers;
5 initialize $\Omega_n = 0$;
6 **for** *i = 0 to I* **do**
7 **for** *each flow* $f \in F$ **do**
8 calculate $z_f = \sum_p y^r_{f,p}$ with $\{y^r_{f,p}\}$;
9 **if** $0 < z_f < 1$ **then**
10 set $z_f = 1$ randomly with the probability of $\epsilon \cdot z_f$;
11 **end**
12 **end**
13 **if** *Eq. (4) is violated* **then**
14 **continue**;
15 **end**
16 calculate $\tilde{\Omega}_d$ with Eq. (7);
17 **if** $\tilde{\Omega}_d \leq \zeta \cdot \Omega_d$ **then**
18 **continue**;
19 **end**
20 invoke *Algorithm* 3 and get $\tilde{\Omega}_n$, $\{\tilde{b}_{u,v}\}$, $\{\tilde{y}_{f,p}\}$ and $\{\tilde{o}_{f,p,i}\}$;
21 **if** $\tilde{\Omega}_n > \Omega_n$ **then**
22 $\Omega_n = \tilde{\Omega}_n$;
23 $\{\bar{b}_{u,v}\} = \{\tilde{b}_{u,v}\}$;
24 $\{y_{f,p}\} = \{\tilde{y}_{f,p}\}$;
25 $\{o_{f,p,i}\} = \{\tilde{o}_{f,p,i}\}$;
26 **end**
27 **end**
28 **return** $\{\bar{b}_{u,v}\}$, $\{y_{f,p}\}$ and $\{o_{f,p,i}\}$.

B. Theoretical Analysis

Theorem 2. *The overall procedure of* Algorithm *2 is polynomial-time.*

Proof. We first consider the time complexity of *Algorithm* 3. In *Algorithm* 3, the outer for-loop covering *Lines* 1-16 will run $|F|$ times. Each time, the inner for-loop covering *Lines* 8-12 will run at most $\max(|P_f|) \cdot (|V| - 1)$ times. In *Lines* 17-20, the for-loop will run $|E|$ times. Hence, the time complexity of *Algorithm* 3 is $O(|F| \cdot |V| \cdot \max(|P_f|) + |E|)$. With regard to *Algorithm* 2, *Lines* 1-4 for solving r-HPR

and r-LLO can be executed in polynomial-time [37], and we denote the time complexity of solving them as O' for convenience. The outer for-loop in *Lines* 6-27 will iterate I times. Each iteration, the inner for-loop in *Lines* 7-12 will run F times. Then the time complexity of the *Algorithm* 2 is $O(O'+I\cdot|F|\cdot|V|\cdot\max(|P_f|)+I\cdot|E|)$. Therefore, the overall procedure of *Algorithm* 2 is polynomial-time. ■

Algorithm 3: Subprocedure of *Algorithm* 2

input : Parameters of *Algorithm* 2, $\{z_f\}$, $\{y^r_{f,p}\}$ and $\{o^r_{f,p,i}\}$.
output: Results of the BaRS problem as $\tilde{\Omega}_n$, $\{\tilde{b}_{u,v}\}$, $\{\tilde{y}_{f,p}\}$ and $\{\tilde{o}_{f,p,i}\}$.

1 **for** *each flow* $f \in F$ **do**
2 **if** $z_f = 1$ **then**
3 find the paths $\tilde{P}_f \subset P_f$ whose propagation delays satisfying τ_f;
4 calculate $\sigma_1 = 1 - \sum_{\tilde{p}\in\tilde{P}_f} y^r_{f,\tilde{p}}$;
5 select a path $\tilde{p}$ from $\tilde{P}_f$ randomly with the probability of $y^r_{f,\tilde{p}} + \frac{\sigma_1}{|\tilde{P}_f|}$;
6 set $\tilde{y}_{f,\tilde{p}} = 1$;
7 set $\tilde{y}_{f,p} = 0$ where $p \in P_f \backslash \tilde{p}$;
8 **for** *each* $o^r_{f,\tilde{p},i}$ **do**
9 obtain the decimal of $o^r_{f,\tilde{p},i}$ as σ_2;
10 set $\tilde{o}_{f,\tilde{p},i} = \lceil o^r_{f,\tilde{p},i} \rceil$ randomly with the probability of σ_2;
11 set $\tilde{o}_{f,\tilde{p},i} = \lfloor o^r_{f,\tilde{p},i} \rfloor$ randomly with the probabilities of $1-\sigma_2$;
12 **end**
13 **else**
14 set $\tilde{y}_{f,p} = 0$ where $p \in P_f$;
15 **end**
16 **end**
17 **for** *each link* $e_{u,v} \in E$ **do**
18 calculate the maximum utilized bandwidth of all cycles in a hypercycle;
19 set $\tilde{b}_{u,v}$ as the minimum feasible value;
20 **end**
21 calculate $\tilde{\Omega}_n$ with Eq. (1);
22 **return** $\tilde{\Omega}_n$, $\{\tilde{b}_{u,v}\}$, $\{\tilde{y}_{f,p}\}$ and $\{\tilde{o}_{f,p,i}\}$.

Lemma 3. *Let* $\{X_i\}$ *be a set of independent random binary variables, and* $S = \sum_i \alpha_i X_i$. *Provided* $\chi_i = \mathbb{E}[X_i]$ *and* $0 < t < \mathbb{E}[S]$, *we have*

$$\mathbb{P}[S \le \mathbb{E}[S] - t] \le \exp\bigg(-\frac{t^2}{2[\sum_i \alpha_i^2(\chi_i - \chi_i^2)]}\bigg). \quad (26)$$

where $\mathbb{P}[\cdot]$ *and* $\mathbb{E}[\cdot]$ *represent probability and mathematical expectation respectively [38], [39].*

Theorem 4. Algorithm *2 can converge to ensure a preset lower bound in the probability of* $\mathbf{P}$.

Proof. We first prove that the algorithm can converge to a lower bound, and then derive the probability $\mathbf{P}$. We introduce a coefficient as ε. With the optimum of the r-HPR as $\hat{\Omega}_n$, we define the probability that *Algorithm* 2 can obtain a solution whose upper-level objective $\tilde{\Omega}_n$ is no more than $\mathbb{E}[\tilde{\Omega}_n] - \varepsilon\cdot\hat{\Omega}_n$ in each iteration as $\mathbf{P}_1 = \mathbb{P}(\tilde{\Omega}_n \le \mathbb{E}[\tilde{\Omega}_n] - \varepsilon\cdot\hat{\Omega}_n)$, where $\mathbb{E}[\tilde{\Omega}_n]$ is the mathematical expectation and $0 < \varepsilon\cdot\hat{\Omega}_n < \mathbb{E}[\tilde{\Omega}_n]$. According to [36], we can have $\mathbf{P}_1 < 1$ if setting ε appropriately. Then, the probability that *Algorithm* 2 can obtain a solution whose upper-level objective $\tilde{\Omega}_n$ is more than $\mathbb{E}[\tilde{\Omega}_n] - \varepsilon\cdot\hat{\Omega}_n$ within I iterations is $\mathbf{P} = 1 - \mathbf{P}_1^I$. Hence, we have $\lim_{I\to+\infty}(1-\mathbf{P}_1^I) = 1$, proving that *Algorithm* 2 can converge to ensure a preset lower bound.

Each iteration in *Algorithm* 2, we can obtain a feasible $\tilde{\Omega}_n$, $\{\tilde{b}_{u,v}\}$, $\{\tilde{y}_{f,p}\}$ and $\{\tilde{o}_{f,p,i}\}$. To calculate $\mathbb{E}[\tilde{\Omega}_n]$, we have

$$\begin{aligned}\mathbb{E}[\tilde{\Omega}_n] &= \mathbb{E}[\tilde{\Omega}_n^1] + \mathbb{E}[\tilde{\Omega}_n^2] \\ &= \sum_{e_{u,v}\in E} b_{u,v} + (\lambda-1)\sum_{e_{u,v}\in E}\mathbb{E}[\tilde{b}_{u,v}] - \\ &\quad \lambda\sum_{f\in F}\sum_{p\in P_f}\sum_{i=0}^{H-1}\frac{\ell_f\cdot Q_{f,i}}{H\cdot c}\cdot|p|\cdot\mathbb{E}[\tilde{y}_{f,p}],\end{aligned} \quad (27)$$

Then, for $\mathbb{E}[\tilde{b}_{u,v}]$, we have

$$\begin{aligned}\mathbb{E}[\tilde{b}_{u,v}] &= \sum_i b_{u,v}\cdot B_i\cdot\mathbb{P}(\tilde{x}_{u,v,i}=1) \\ &= b_{u,v}\cdot B_0\cdot\mathbb{P}(R_{u,v,j} \le b_{u,v}\cdot B_0\cdot c) + \\ &\quad \sum_{i>0} b_{u,v}\cdot B_i\cdot\mathbb{P}(b_{u,v}\cdot B_{i-1}\cdot c \le R_{u,v,j} \le b_{u,v}\cdot B_i\cdot c),\end{aligned} \quad (28)$$

where $R_{u,v,j}$ denotes the amount of data in cycle j on link $e_{u,v}$. To derive $\mathbb{P}(\cdot)$ in Eq. (28), we define $\tilde{R}^{\max}_{u,v}$ and $\tilde{R}^{\min}_{u,v}$ as a maximum and a minimum estimation of R respectively, which can be estimated as

$$\tilde{R}^{\max}_{u,v} = \sum_{f,p}\tilde{y}_{f,p}\cdot\max_j(\ell_f\cdot Q_{f,j}), \quad \forall e_{u,v},$$

$$\tilde{R}^{\min}_{u,v} = \sum_{f,p}\tilde{y}_{f,p}\cdot\min_j(\ell_f\cdot Q_{f,j}), \quad \forall e_{u,v}.$$

Then for link $e_{u,v}$, we have

$$\begin{aligned}\mathbb{P}(R_{u,v,j} \le b_{u,v}\cdot B_0\cdot c) &\ge \mathbb{P}(\tilde{R}^{\max}_{u,v} \le b_{u,v}\cdot B_0\cdot c) \\ &= \mathbf{P}_2.\end{aligned} \quad (29)$$

To calculate $\mathbf{P}_2$, we consider the problem in **Definition 2**.

Definition 2: Assume there are a set of items, each with a weight as $\max_j(\ell_f\cdot Q_{f,j})$. We select some from the items such that the total weight of the selected ones is no more than $b_{u,v}\cdot B_0\cdot c$. If we use $\tilde{y}^*_{f,p} = 1$ to denote that an item is selected and 0 otherwise, a feasible selection can be written as $\{\tilde{y}^*_{f,p}, \forall f,p\}$. The problem is to figure out all feasible selections.

The problem in **Definition 2** actually constitutes the well-known 0-1 knapsack problem, and can be solved with dynamic programming, during which all feasible solutions can be collected. Then back to Eq. (29), we have

$$\mathbf{P}_2 = \sum_{\{\tilde{y}^*_{f,p}\}}\prod_{\tilde{y}^*_{f,p}}\hat{y}^*_{f,p} \quad (30)$$

where $\hat{y}^*_{f,p}$ is defined as

$$\hat{y}^*_{f,p} = \begin{cases} \tilde{y}^r_{f,p}, & \tilde{y}^*_{f,p} = 1, \\ 1 - \tilde{y}^r_{f,p}, & \tilde{y}^*_{f,p} = 0. \end{cases}$$

Similarly, for link $e_{u,v}$, we have

$$\begin{aligned} &\mathbb{P}(b_{u,v} \cdot B_{i-1} \cdot c \le R_{u,v,j} \le b_{u,v} \cdot B_i \cdot c) \\ &= 1 - \mathbb{P}(R_{u,v,j} \ge b_{u,v} \cdot B_i \cdot c) - \mathbb{P}(R_{u,v,j} \le b_{u,v} \cdot B_{i-1} \cdot c) \\ &\ge 1 - \mathbb{P}(\tilde{R}^{\max}_{u,v} \ge b_{u,v} \cdot B_i \cdot c) - \mathbb{P}(\tilde{R}^{\min}_{u,v} \le b_{u,v} \cdot B_{i-1} \cdot c) \\ &= 1 - \mathbf{P}_3 - \mathbf{P}_4. \end{aligned} \tag{31}$$

To calculate $\mathbf{P}_3$ and $\mathbf{P}_4$, we consider the problems in **Definition 3** and **Definition 4**, which are similar to **Definition 2**.

Definition 3: Assume there are a set of items, each with a weight as $\max_i(\ell_f \cdot Q_{f,i})$. We select some from the items such that the total weight is no less than $b_{u,v} \cdot B_i \cdot c$. If we use $\tilde{y}^\diamond_{f,p} = 1$ to denote that an item is selected and 0 otherwise, a feasible solution can be written as $\{\tilde{y}^\diamond_{f,p}, \forall f, p\}$. The problem is to figure out all feasible selections.

Definition 4: Assume there are a set of items, each with a weight as $\min_i(\ell_f \cdot Q_{f,i})$. We select some from the items such that the total weight is no more than $b_{u,v} \cdot B_{i-1} \cdot c$. If we use $\tilde{y}^o_{f,p} = 1$ to denote that an item is selected and 0 otherwise, a feasible solution can be written as $\{\tilde{y}^o_{f,p}, \forall f, p\}$. The problem is to figure out all feasible selections.

Then back to Eq. (31), we have $\mathbf{P}_3$ as

$$\mathbf{P}_3 = \sum_{\{\tilde{y}^\diamond_{f,p}\}} \prod_{\tilde{y}^\diamond_{f,p}} \hat{y}^\diamond_{f,p} \tag{32}$$

where $\hat{y}^\diamond_{f,p}$ is defined as

$$\hat{y}^\diamond_{f,p} = \begin{cases} \tilde{y}^r_{f,p}, & \tilde{y}^\diamond_{f,p} = 1, \\ 1 - \tilde{y}^r_{f,p}, & \tilde{y}^\diamond_{f,p} = 0, \end{cases}$$

and have $\mathbf{P}_4$ as

$$\mathbf{P}_4 = \sum_{\{\tilde{y}^o_{f,p}\}} \prod_{\tilde{y}^o_{f,p}} \hat{y}^o_{f,p} \tag{33}$$

where $\hat{y}^o_{f,p}$ is defined as

$$\hat{y}^o_{f,p} = \begin{cases} \tilde{y}^r_{f,p}, & \tilde{y}^o_{f,p} = 1, \\ 1 - \tilde{y}^r_{f,p}, & \tilde{y}^o_{f,p} = 0. \end{cases}$$

Then with Eq. (29) and Eq. (31), $\mathbb{E}[\tilde{b}_{u,v}]$ in Eq. (28) can be written as

$$\mathbb{E}[\tilde{b}_{u,v}] \ge b_{u,v} \cdot B_0 \cdot \mathbf{P}_2 + \sum_{i>0} b_{u,v} \cdot B_i \cdot (1 - \mathbf{P}_3 - \mathbf{P}_4). \tag{34}$$

Back to Eq. (27), we have

$$\begin{aligned} \mathbb{E}[\tilde{\Omega}_n] &= \mathbb{E}[\tilde{\Omega}^1_n] + \mathbb{E}[\tilde{\Omega}^2_n] \\ &\ge \sum_{e_{u,v} \in E} b_{u,v} + (\lambda - 1) \cdot \\ &\quad \sum_{e_{u,v} \in E} (b_{u,v} \cdot B_0 \cdot \mathbf{P}_2 + \sum_{i>0} b_{u,v} \cdot B_i \cdot (1 - \mathbf{P}_3 - \mathbf{P}_4)) \\ &\quad - \lambda \sum_{f \in F} \sum_{p \in P_f} \sum_{i=0}^{H-1} \frac{\ell_f \cdot Q_{f,i}}{H \cdot c} \cdot |p| \cdot \tilde{y}^r_{f,p}, \end{aligned} \tag{35}$$

According to **Lemma** 3, we set $\{\alpha_i\} = \Big\{\overbrace{..., (\lambda-1) \cdot b_{u,v} \cdot B_i, ...}^{|E||B|}, ..., \overbrace{\frac{\lambda \cdot \ell_f \cdot Q_{f,i}}{H \cdot c} \cdot |p|, ...}^{H \sum_f |P_f|}\Big\}$, $\{\chi_i\} = \big\{\{\mathbb{P}[\tilde{x}_{u,v,i} = 1],\ \forall e_{u,v} \in E, i \in [0, |B|)\}, \{\tilde{y}^r_{f,p},\ \forall f \in F, p \in P_f\}\big\}$, and $t = \varepsilon \cdot \hat{\Omega}_n$. Then,

$$\mathbb{P}[\tilde{\Omega}_n \le E[\tilde{\Omega}_n] - \varepsilon \cdot \hat{\Omega}_n] \le \exp\Big(- \frac{(\varepsilon \cdot \hat{\Omega}_n)^2}{2 \sum_i \alpha_i^2 \chi_i} \Big) = \mathbf{P}_1,$$

where $E[\tilde{\Omega}_n]$, $\hat{\Omega}_n$ and $\sum_i \alpha_i^2 \chi_i$ can be obtained with *Algorithm* 2. Finally, the probability $\mathbf{P}$ that *Algorithm* 2 can converge to ensure a preset lower bound $E[\tilde{\Omega}_n] - \varepsilon \cdot \hat{\Omega}_n$ is at least $1 - \mathbf{P}^I_1$. ■

VII. Performance Evaluation

In this section, we perform both small-scale and large-scale simulations to evaluate the performance of the proposed algorithms.

A. Simulation Setup

We adopt two real-world network topologies for simulations, i.e., Netrail and Sprint. The topologies are shown in Fig. 4, where link lengths are provided in kilometers. In each topology, each egress port of nodes has $N = 3$ queues dedicated for deterministic traffic. Each link has a bandwidth of 10 Gbps and the set of possible portions of bandwidth on each link is assumed as $B = [20\%, 40\%, 60\%, 80\%]$. We take link lengths as weights and precompute at most 5 shortest paths for each pair of nodes. Temporally, we assume that each hyper-cycle consists of 12 cycles and each cycle has a duration of 10 μs.

Under each topology, deterministic flows are generated. For each flow, its source and destination are randomly selected. Its maximum tolerable end-to-end delay is generated randomly to be larger than the smallest propagation delay from the source to destination, which ensures that any flow has at least a candidate path. All flows have their packet sizes distributed within [64, 1500] Bytes, where 30% of them have their packet sizes fixed as 64 Bytes, 30% of them have their packet sizes fixed as 1500 Bytes and the others have their packet sizes randomly selected from (64, 1500) Bytes. We assume that the traffic of all flows repeats per hyper-cycle. During each cycle, each flow can have either some packets or no packet to sent. In case there are some packets to be sent, we set the number of packets as 1 or 2 randomly, corresponding to a maximum traffic of 2.4 Gbps.

We evaluate the performance of the bilevel optimization by comparing it with two single-level optimizations, which are the one that only optimizes the lower-level objective and the one that only optimizes the upper-level objective. Specifically, the first one is adapted from [9] with bandwidth allocated

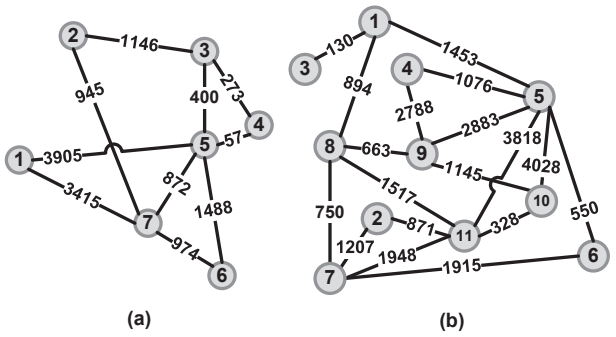

Fig. 4. Network topologies (link lengths are marked in kilometers): (a) Netrail, (b) Sprint.

in a one-size-fits-all approach, and can be formulated with Eqs. (8)-(16). The second one can be formulated with Eq. (1), Eqs. (2)-(7) and Eqs. (9)-(16). In the following, we use BLO, SLO-d and SLO-n to represent the three optimizations for simplicity. For BLO and SLO-n, we set $\lambda = 0.3$ in Eq. (1) to weigh Ω_n^1 and Ω_n^2. Our simulations are implemented in a Python environment with DOcplex [40] and then run on virtual machines with 2.5/3.2 GHz 8 vCPU and 32 GB RAM. To reduce statistical fluctuations, we average over 5 independent traffic realizations to get each data point of the simulation results.

B. Small-scale Simulations

The small-scale simulations use the Netrail topology to compare the performance of four algorithms, which are 1) the one that BLO is solved exactly with *Algorithm* 1 (BLO/Exact), 2) the one that SLO-n is solved exactly with Cplex (SLO-n/Exact), 3) the one that SLO-d is solved exactly with Cplex (SLO-d/Exact), and 4) the one that BLO is solved approximately with *Algorithm* 2 (BLO/Approx), where we perform iterations to ensure that the relative gap between $\tilde{\Omega}_n$ and $\hat{\Omega}_n$ is less than 10%. Due to the time complexity of exact algorithms, we set the number of deterministic flows to be provisioned as 6, 9 and 12 respectively. In this scenario, we also change the ratio of accepted deterministic throughput from $\rho = 100\%$ to $\rho = 80\%$ to explore its effects.

Table IV summarizes the simulation results of the four algorithms of Ω_n, the number of accepted flows, Ω_n^1, and the running time. Note that to save space, we only show the results of SLO-d/Exact obtained under the circumstance that 60% of bandwidth is allocated on each link for deterministic flows. And we have confirmed that under the circumstances, the performance in balancing Ω_n and the number of accepted flows is the best when compared with the other three circumstances where 20%, 40% and 80% of bandwidth are allocated. As expected, SLO-n/Exact achieves the largest Ω_n while SLO-d/Exact achieves the largest number of accepted flows. Comparing the results on Ω_n, we can see that no matter what the number of total flows and ρ are, the performances of BLO/Exact and BLO/Approx always approach that of SLO-n/Exact, and are always much better than that of SLO-d/Exact. From the results on the number of accepted flows, we can observe that both BLO/Exact and BLO/Approx achieve similar performance to SLO-d/Exact, and perform the same as or better than SLO-n/Exact. Furthermore, with ρ decreasing, the performance gap between SLO-n/Exact and BLO/Exact or BLO/Approx widens. These results verify the correctness and effectiveness of BLO/Exact and BLO/Approx in balancing the tradeoff between the upper-level and the lower-level objectives, which benefits from the two-level optimization structure. Among the four algorithms, the running time of BLO/Exact is the longest, and moreover increases rapidly with ρ decreasing, which confirms the necessity of proposing the approximation algorithm.

Tables V-VI list the detailed results of BaRS of the four algorithms under $\rho = 100\%$ and $\rho = 80\%$ respectively, which are obtained manually by analyzing the simulations. To save space, we only show the results of a group of 6 flows, which to the largest extent exhibit the differences among the algorithms and hence can help analyze the algorithms better. Under $\rho = 100\%$, we can see that compared with SLO-d/Exact, the other three algorithms implement flexible and compact bandwidth allocation to provision deterministic traffic and therefore will produce more opportunities for normal traffic. When $\rho = 80\%$, BLO/Exact and BLO/Approx can still provision deterministic traffic as much as possible as the lower-level optimization directs, while SLO-n/Exact provision just enough deterministic traffic. These results explicitly suggest that BLO/Exact and BLO/Approx always produce similar solutions, but different solutions from those of SLO-n/Exact and SLO-d/Exact.

C. Large-scale Simulations

We perform large-scale simulations with Sprint. Due to the time complexity of exact algorithms, we compare the performances of approximation algorithms, i.e., BLO/Approx, SLO-n/Approx and SLO-d/Approx, where the latter two correspond to solving the SLO-n and SLO-d optimizations with LPR and randomized rounding whose procedures are similar to *Algorithm* 2 and the algorithm in [9] respectively. Specifically for SLO-d/Approx, we set the bandwidth allocation portions as those in B, and denote the algorithm counterparts as SLO-d/Approx/20%, SLO-d/Approx/40%, SLO-d/Approx/60% and SLO-d/Approx/80% respectively. As proved in **Theorem** 4, randomized rounding can converge to ensure a preset lower bound in certain probability, and the more iterations the higher probability. For time-efficiency and fair comparison, we prepare the LPRs in advance, and conduct the same number of iterations (i.e., 10^3) for rounding in all the algorithms. In this scenario, we change the number of flows within the range [100, 400], and set ρ as a small value of 20% to better reveal the efficiency of BLO.

Figs. 5-7 provide the results of Ω_n, Ω_n^1, and the results of the average ratio of accepted deterministic flows, respectively. These results further verify that BLO/Approx can balance the tradeoff between the upper-level and lower-level objectives than the benchmarks. Specifically, since SLO-n/Approx concentrates on optimizing the upper-level objective, it achieves higher Ω_n than BLO/Approx as Fig. 5 illustrates. However, it also results in much lower average ratio of accepted deterministic flows than BLO/Approx. As for SLO-d/Approx,

TABLE IV
RESULTS OF SMALL-SCALE SIMULATIONS.

Metrics		T_n (Gbps)			T_n^1 (Gbps)			Average Accepted Flows			Running Time (s)		
Total Flows		6	9	12	6	9	12	6	9	12	6	9	12
$\rho = 100\%$	BLO/Exact	165.139	161.101	160.241	153.200	148.800	148.800	6.000	9.000	12.000	3.538	7.392	11.879
	SLO-n/Exact	165.139	161.101	160.241	153.200	148.800	148.800	6.000	9.000	12.000	2.167	4.706	6.482
	BLO/Approx	164.865	160.693	159.349	152.800	148.000	147.600	6.000	9.000	12.000	3.116	8.320	12.948
$\rho = 90\%$	BLO/Exact	165.142	161.101	160.605	153.200	148.800	149.200	6.000	9.000	11.800	10.931	57.737	284.442
	SLO-n/Exact	165.325	161.419	161.009	153.200	148.800	149.200	4.200	5.600	8.800	2.029	5.068	7.269
	BLO/Approx	164.753	160.689	160.113	152.800	148.000	148.400	6.000	9.000	11.800	3.719	8.570	13.450
$\rho = 80\%$	BLO/Exact	165.142	164.387	163.465	153.200	152.400	152.400	6.000	8.000	11.000	15.591	21.655	198.145
	SLO-n/Exact	165.478	164.450	164.004	153.200	152.400	152.800	3.200	6.200	8.400	1.936	4.747	8.355
	BLO/Approx	164.724	160.693	160.109	152.800	148.000	148.400	6.000	9.000	11.800	4.043	7.186	13.304
SLO-d/Exact		113.719	112.567	111.627	80.000	80.000	80.000	6.000	9.000	12.000	1.545	3.169	5.389

TABLE V
BARS RESULTS OF 6 FLOWS UNDER $\rho = 100\%$.

Results	BLO/Exact	SLO-n/Exact	SLO-d/Exact	BLO/Approx
Bandwidth Allocation	20%: All links\[1→7, 3→5], 40%: [1→7, 3→5], 60%: None, 80%: None	20%: All links\[1→7, 3→5], 40%: [1→7, 3→5], 60%: None, 80%: None	20%: None, 40%: None, 60%: All links, 80%: None	20%: All links\[1→7, 3→5, 5→3] 40%: [1→7, 3→5, 5→3], 60%: None, 80%: None
Routing	f_1: 5→7→2→3, f_2: 1→7 f_3: 3→5, f_4: 4→5→6 f_5: 5→3→2, f_6: 5→7	f_1: 5→7→2→3, f_2: 1→7 f_3: 3→5, f_4: 4→5→6 f_5: 5→3→2, f_6: 5→7	f_1: 5→7→2→3, f_2: 1→7 f_3: 3→5, f_4: 4→5→6 f_5: 5→3→2, f_6: 5→7	f_1: 5→3, f_2: 1→7 f_3: 3→5, f_4: 4→5→6 f_5: 5→3→2, f_6: 5→7
Scheduling	f_1: [1, 1, 1], f_2: [0] f_3: [0], f_4: [1, 0] f_5: [0, 0], f_6: [0]	f_1: [1, 1, 1], f_2: [0] f_3: [0], f_4: [1, 0] f_5: [0, 0], f_6: [0]	f_1: [0, 0, 0], f_2: [0] f_3: [0], f_4: [0, 1] f_5: [1, 1], f_6: [0]	f_1: [1], f_2: [1] f_3: [0], f_4: [1, 1] f_5: [1, 1], f_6: [1]

Ω_n decreases but the average ratio of accepted deterministic flows increases with more bandwidth allocated for deterministic traffic. When compared with SLO-d/Approx, BLO-d/Approx achieves comparable Ω_n to SLO-d/Approx/40%. But it admits more deterministic flows into the network than SLO-d/Approx/40%, and as Fig. 6 shows, the average ratios of accepted deterministic flows even reach to that of SLO-d/Approx/60%. For example, when the numbers of flows are 100 and 150, the ratios are almost the same as those of SLO-d/Approx/60%, and approximately 10% higher than those of SLO-d/Approx/40%. Fig. 7 also provides the results of Ω_n^1, which suggests that such performance of BLO/Approx over SLO-n/Approx and SLO-d/Approx arises from both bandwidth allocation and routing and scheduling. To analyze the convergence performance, we fix the number of flows as 100 and plot the results of the upper-level objective under all 5 traffic realizations for BLO/Approx. As Fig. 8 shows, the upper-level objective exhibits rapid improvement within the initial hundreds of iterations and gets converged with more iterations, which numerically confirms **Theorem 4**. The results of running time are shown in Table VII, where the algorithms consume time in similar time-scale.

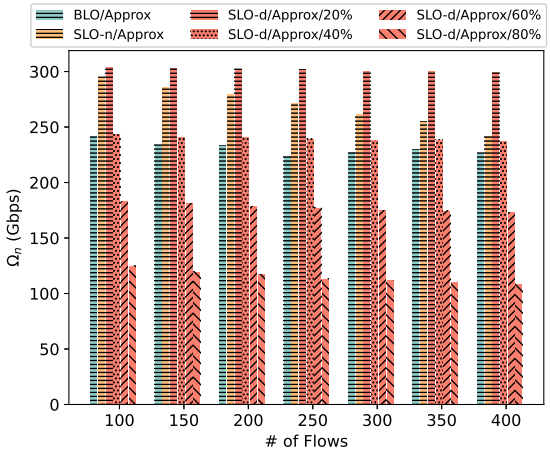

Fig. 5. Results of Ω_n.

VIII. CONCLUSION

In this work, we studied the BaRS problem, and modelled this problem as a bilevel optimization which consists of an upper-level optimization and a lower-level optimization. The upper-level optimization addressed bandwidth allocation between deterministic and normal traffic to maximize the bandwidth available for normal traffic; the lower-level opti-

TABLE VI
BARS RESULTS OF 6 FLOWS UNDER $\rho = 80\%$

Results	BLO/Exact	SLO-n/Exact	SLO-d/Exact	BLO/Approx
Bandwidth Allocation	20%: All links\[1→7, 3→5], 40%: [1→7, 3→5], 60%: None, 80%: None	20%: All links\[1→7, 3→5], 40%: [1→7, 3→5], 60%: None, 80%: None	20%: None 40%: None 60%: All links, 80%: None	20%: All links\[1→7, 3→5], 40%: [1→7, 3→5], 60%: None, 80%: None
Routing	f_1: 5→4→3, f_2: 1→7 f_3: 3→5, f_4: 4→5→6 f_5: 5→3→2, f_6: 5→7	f_1: None, f_2: 1→7 f_3: 3→5, f_4: None f_5: 5→3→2, f_6: 5→7	f_1: 5→7→2→3, f_2: 1→7 f_3: 3→5, f_4: 4→5→6 f_5: 5→3→2, f_6: 5→7	f_1: 5→3, f_2: 1→7 f_3: 3→5, f_4: 4→5→6 f_5: 5→4→3→2, f_6: 5→7
Scheduling	f_1: [1, 1], f_2: [0] f_3: [0], f_4: [1, 1] f_5: [0, 0], f_6: [0]	f_1: None, f_2: [0] f_3: [0], f_4: None f_5: [0, 0], f_6: [1]	f_1: [0, 0, 0], f_2: [0] f_3: [0], f_4: [0, 1] f_5: [1, 1], f_6: [0]	f_1: [1], f_2: [1] f_3: [0], f_4: [1, 1] f_5: [1, 1, 1], f_6: [1]

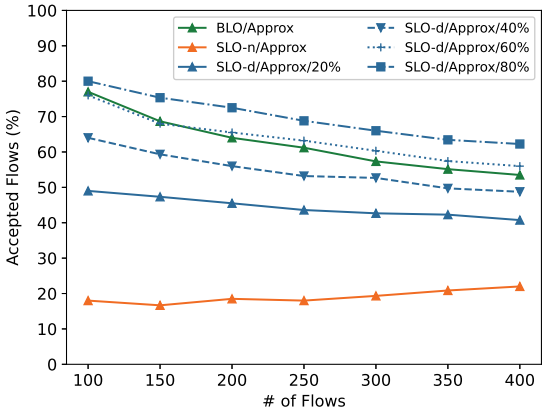

Fig. 6. Results of average ratio of accepted to total deterministic flows.

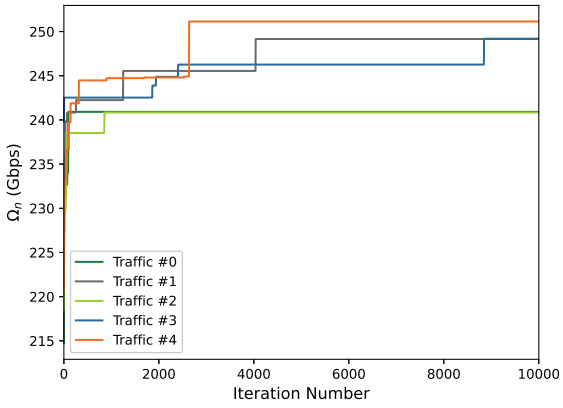

Fig. 8. Results of convergence performance for BLO/Approx under 100 flows.

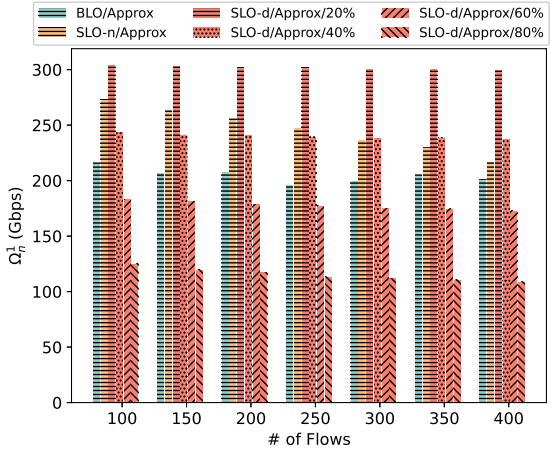

Fig. 7. Results of Ω_n^1.

TABLE VII
RUNNING TIME (SECONDS) IN LARGE-SCALE SIMULATIONS.

Total Flows	BLO/Approx	SLO-n/Approx	SLO-d/Approx/·
100	53.338	19.672	42.006
150	110.282	35.774	96.236
200	179.091	66.982	152.388
250	255.434	105.620	233.466
300	355.822	150.246	332.924
350	446.564	262.115	450.188
400	568.810	281.366	554.915

mization addressed RS for deterministic traffic to maximize the number of accepted flows. We first formulated the bilevel optimization as a BMILP. Then, we proposed an exact algorithm based on cutting planes to solve it exactly, and proposed an approximation algorithm based on two-level relaxations and randomized rounding to solve it time-efficiently. Extensive simulations were conducted and the results verified the effectiveness of our proposals in balancing the tradeoff between the bandwidth available for normal traffic and the number of accepted deterministic flows.

ACKNOWLEDGMENTS

This work was supported by the Science and Technology Foundation of the State Grid Corporation of China (5700-202455369A-3-1-DG).

REFERENCES

[1] I. Parvez, A. Rahmati, I. Guvenc, A. Sarwat, and H. Dai, "A survey on low latency towards 5g: Ran, core network and caching solutions," *IEEE Commun. Surveys Tut.*, vol. 20, no. 4, pp. 3098–3130, 2018.

[2] E. Grossman, "Deterministic networking use cases," *Internet Engineering Task Force*, 2019.

[3] N. Finn, P. Thubert, B. Varga, and J. Farkas, "Deterministic networking architecture," *RFC 8655*, 2019.

[4] M. Chen, X. Geng, Z. Li, J. Joung, and J. Ryoo, "Segment Routing (SR) Based Bounded Latency," *Internet Engineering Task Force*, Jul. 2023.

[5] S. Peng, P. Liu, K. Basu, A. Liu, D. Yang, and G. Peng, "Timeslot Queueing and Forwarding Mechanism," *Internet Engineering Task Force*, Nov 2024.

[6] S. Peng, Z. Du, K. Basu, Z. Cheng, D. Yang, and C. Liu, "Deadline Based Deterministic Forwarding," *Internet Engineering Task Force*, Mar 2025.

[7] "Ne40e series universal service routers," [Online]. Available: https://support.huawei.com/enterprise/en/routers/ne40e-pid-15837.

[8] "Cr16000-f high-end router series," [Online]. Available: https://www.h3c.com/en/Products-and-Solutions/InterConnect/Routers/Products/WAN-Routers/CR/H3C-CR16000-F/.

[9] J. Krolikowski, S. Martin, P. Medagliani, J. Leguay, S. Chen, X. Chang, and X. Geng, "Joint routing and scheduling for large-scale deterministic ip networks," *Comput. Commun.*, vol. 165, pp. 33–42, 2021.

[10] W. Wu, X. Zhang, J. Pan, and Y. Zhou, "Joint optimization of time-slot allocation and traffic steering for large-scale deterministic networks," *J. Commun. Netw.*, vol. 25, no. 6, pp. 825–840, 2023.

[11] G. P. Sharma, W. Tavernier, D. Colle, and M. Pickavet, "Routing and scheduling for 1+ 1 protected detnet flows," *Comput. Netw.*, vol. 211, p. 108960, 2022.

[12] M. Wang, S. Zhu, L. Wang, W. Li, and Y. Zhang, "Heuristic fast routing in large-scale deterministic network," in *Proc. of IPCCC 2023*, Nov. 2023, pp. 9–17.

[13] H. Yu, T. Taleb, and J. Zhang, "Deep Reinforcement Learning-Based Deterministic Routing and Scheduling for Mixed-Criticality Flows," *IEEE Trans. Ind. Informat.*, vol. 19, pp. 8806–8816, 2023.

[14] Z. Quan and J.-M. Chung, "Priority queueing analysis of self-similar in high-speed networks," in *Proc. of ICC 2003*, 2003, pp. 1606–1610.

[15] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose atm switch chip," *IEEE J. Sel. Areas Commun.*, vol. 9, pp. 1265–1279, 1991.

[16] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Trans. Netw.*, vol. 4, pp. 375–385, 1996.

[17] J. Bennett and H. Zhang, "Wf/sup 2/q: worst-case fair weighted fair queueing," in *Proc. of INFOCOM 1996*, 1996, pp. 120–128.

[18] "Forwarding and queuing enhancements for time-sensitive streams," *IEEE Std 802.1Qav-2009*, pp. 1–72, 2010.

[19] "Enhancements for scheduled traffic," *IEEE Std 802.1Qbv-2015*, pp. 1–57, 2016.

[20] "Cyclic queuing and forwarding," *IEEE 802.1Qch-2017*, pp. 1–30, 2017.

[21] L. Qiang, X. Geng, B. Liu, T. Eckert, L. Geng, and G. Li, "Large-scale deterministic ip network," *Internet Engineering Task Force*, 2019.

[22] J. Yan, W. Quan, X. Jiang, and Z. Sun, "Injection time planning: Making cqf practical in time-sensitive networking," in *Proc. of INFOCOM 2020*, 2020, pp. 616–625.

[23] Y. Zhang, Q. Xu, L. Xu, C. Chen, and X. Guan, "Efficient flow scheduling for industrial time-sensitive networking: A divisibility theory-based method," *IEEE Trans. Ind. Informat.*, vol. 18, pp. 9312–9323, 2022.

[24] X. Wang, H. Yao, T. Mai, Z. Xiong, F. Wang, and Y. Liu, "Joint routing and scheduling with cyclic queuing and forwarding for time-sensitive networks," *IEEE Trans. Veh. Technol.*, vol. 72, pp. 3793–3804, 2023.

[25] D. Yang, Z. Cheng, W. Zhang, H. Zhang, and X. Shen, "Burst-aware time-triggered flow scheduling with enhanced multi-cqf in time-sensitive networks," *IEEE/ACM Trans. Netw.*, vol. 31, pp. 2809–2824, 2023.

[26] X. He, X. Zhuge, F. Dang, W. Xu, and Z. Yang, "Deepscheduler: Enabling flow-aware scheduling in time-sensitive networking," in *Proc. of INFOCOM 2023*, 2023, pp. 1–10.

[27] Y. Huang, S. Wang, X. Zhang, T. Huang, and Y. Liu, "Flexible cyclic queuing and forwarding for time-sensitive software-defined networks," *IEEE Trans. Netw. Serv.*, vol. 20, pp. 533–546, 2023.

[28] D. Yang, W. Zhang, Q. Ye, C. Zhang, N. Zhang, C. Huang, H. Zhang, and X. Shen, "Detfed: Dynamic resource scheduling for deterministic federated learning over time-sensitive networks," *IEEE Trans. Mob. Comput.*, vol. 23, no. 5, pp. 5162–5178, 2024.

[29] Q. Lv, F. Zhou, and Z. Zhu, "On the bilevel optimization to design control plane for SDONs in consideration of planned physical-layer attacks," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 3221–3230, 2020.

[30] I. Bouras, R. Figueiredo, M. Poss, and F. Zhou, "Minimizing energy and link utilization in isp backbone networks with multi-path routing: a bi-level approach," *Optim. Lett.*, vol. 14, no. 1, pp. 209–227, 2020.

[31] H. Yang, X. Pan, S. Zhao, B. Ge, H. Yu, and Z. Zhu, "On the bilevel optimization for remapping virtual networks in an hoe-dcn," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 2, pp. 1274–1286, 2022.

[32] Nasrallah, A. Thyagaturu, A. S. Alharbi, Z. Wang, C. Shao, X. Reisslein, and M. E. Hesham, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Commun. Surveys Tut.*, vol. 21, no. 1, pp. 88–145, 2018.

[33] T. Kleinert, M. Labbé, I. Ljubić, and M. Schmidt, "A survey on mixed-integer programming techniques in bilevel optimization," *EURO J. Comput. Optim.*, vol. 9, p. 100007, 2021.

[34] "Cplex optimization studio," [Online]. Available: https://www.ibm.com/products/ilog-cplex-optimization-studio.

[35] "Gurobi," [Online]. Available: https://www.gurobi.com.

[36] T. C. Raghavan, P., "Randomized rounding: A technique for provably good algorithms and algorithmic proofs," *Combinatorica*, vol. 7, no. 8, pp. 365–374, 1987.

[37] A. Schrijver, "Theory of linear and integer programming," *Comput. Math. Appl.*, vol. 36, no. 8, pp. 122–593, 1998.

[38] S. Boucheron, G. Lugosi, and P. Massart, *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[39] N. Bansal, "On a generalization of iterated and randomized rounding," in *Proc. of SIGACT 2019*, 2019, pp. 1125–1135.

[40] "Ibm decision optimization cplex modeling for python," [Online]. Available: https://github.com/IBMDecisionOptimization/docplex-examples.