E2E Latency-Bounded Routing with SLA Guarantee under MaxWeight Scheduling

Dawson Berry[1] and Jie Gao[1, 2]
[1]Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada
[2]School of Information Technology, Carleton University, Ottawa, ON, Canada
Emails: {dawsonberry@cmail.carleton.ca, jie.gao6@carleton.ca}

***Abstract*—This paper investigates joint admission control, routing, and per-hop queue-weight provisioning for latency-sensitive flows in networks employing per-hop MaxWeight (MW) scheduling. To overcome the absence of closed-form queuing laws, we introduce an analytically tractable per-queue slotted-time model and use it to derive a provable tail bound on end-to-end latency. Based on this bound, we propose a method to derive the required per-hop service share for a given service-level agreement (SLA) and map it to queue-weight parameters, ensuring that each queue maintains its provisioned share under MW scheduling. Finally, we design a branch-and-bound algorithm to find optimal loop-free paths over sequences of virtual queues. Simulation results demonstrate that our approach achieves zero SLA violations by trading off throughput for SLA compliance, highlighting its potential for time-sensitive networking scenarios.**

***Index Terms*—Latency-bounded routing, virtual output queues, MaxWeight, time-sensitive networking, input-queued switch**

I. Introduction

Meeting stringent end-to-end (E2E) latency requirements with near-zero violation probability is a fundamental challenge in cloud–edge networks supporting latency-critical applications such as extended reality (XR) and industrial automation [1], [2]. While time-sensitive networking (TSN) and other QoS mechanisms can reduce and stabilize per-hop delay, meeting E2E SLAs for multiple flows necessitates judicious resource allocation and effective resolution of contention over shared underlying resources.

Contention for per-hop resources is typically resolved locally at each hop by a packet scheduler. A widely used throughput-optimal per-hop scheduling scheme is MaxWeight (MW), which matches input queues to output ports in each time slot so as to maximize the sum of scheduled queue weights [3], [4]. However, MW is not readily amenable to routing design with E2E latency guarantees, due to the lack of tractable closed-form characterizations of its delay performance [5]. Routing with E2E latency guarantees requires the knowledge of per-hop delay behavior and, therefore, analytically tractable queuing abstractions to model per-hop sojourn times under MW scheduling.

One option is to abstract each hop as an independent first-in, first-out (FIFO) server using queuing models with closed forms, such as $M/M/1$ and $M/D/1$. However, this fails to capture three key characteristics of input-queued (IQ) switches, including matching contention among input queues (i.e., at most one input may be matched to a given output at a time), queue-length-dependent service under scheduling (i.e., the service opportunity of a queue often depends on its backlog), and input–output coupling in packet latency (i.e., the sojourn time of a packet depends on both the input and the output queues). An alternative option of abstraction is to model a switch as a network of independent FIFO servers, each with an individual queue. However, since the scheduler chooses which queues may receive service and process packets, its choices tie the queues together, rendering isolated analyses of queuing delay, state evolution, and busy periods inaccurate.

Both of the aforementioned abstraction options can lead to issues such as head-of-line (HOL) blocking, inter-flow interference driven by aggregate traffic rather than scheduling decisions, and violations of link-capacity or input–output constraints. Moreover, even with an accurate closed-form model of per-hop sojourn time, quality of service (QoS) routing with multiple constraints, such as an end-to-end delay target, resource feasibility, and a loop-free path requirement, is an instance of multi-constrained path (MCP) selection, which is NP-complete [6].

To address these challenges, we propose an approach with three components. First, we model the slotted MW scheduling service process as an i.i.d. Bernoulli process, based on which closed-form expressions for a downstream arrival process are derived. These closed forms are then used to invert an E2E-latency tail bound and obtain a required per-hop service share, defined as a conditional service probability for a backlogged virtual output queue (VOQ). Second, we map the required service share to queue weights, which are dynamically updated upon the admission of new flows to preserve the SLAs of existing flows. Third, we develop a routing algorithm that operates on the input–output queue abstraction and computes end-to-end routes as sequences of virtual queues along a feasible path. The contributions of this work include (i) transforming an E2E latency budget into a single tunable parameter; (ii) a finite monotonic queue-weight search algorithm for SLA satisfaction; and (iii) a lightweight routing algorithm for computing loop-free and latency-bounded paths via minimum queue-weight increases. The proposed approach achieves zero SLA violations in simulation under all evaluated regimes, trading off throughput through conservative flow admission.

II. Related Work

MW scheduling is throughput-optimal in the presence of resource contention [3]. For generalized switches, heavy-traffic analyses provide insights such as state-space collapse and workload minimization under MW [4]. These results characterize stability regions, queue-length scaling, and asymptotic optimality. However, MW can exhibit undesirable delay coupling across flows under heavy-tailed traffic, which can propagate delay instability. Several lines of work have focused on providing bounded-delay service within a single switch. One approach assumes prior knowledge of the input–output traffic rate matrix, which allows the resulting scheduling policy to yield analytically tractable delay guarantees for reserved traffic [7]. Techniques such as leaky-bucket regulation and departure-time bounds can be applied at individual switches to achieve bounded delay [8]. However, at the network level, techniques that perform well for an isolated switch may fail to provide E2E latency guarantees across a network of switches, which has motivated algorithms and stability analyses for networks of IQ switches [9], [10].

QoS routing often corresponds to an MCP problem, motivating heuristic algorithms such as LARAC [11]. Online joint flow admission and routing problems are often solved using interference-aware or flow-profile-aware methods such as MIRA [12]. Deterministic network calculus (DNC) provides worst-case bounds on delay and backlog using arrival and service curves, and is widely used for timing analysis and verification in TSN and deterministic networking (DetNet). However, computing tight worst-case bounds using DNC often leads to prohibitive computational complexity. Moreover, even when such bounds can be computed, they can be conservative due to the gap between abstract service models and actual switch behavior [13], [14]. Finally, the joint assignment of per-node shaper and queue parameters under an end-to-end delay constraint constitutes an NP-hard optimization problem [15].

Therefore, prior research does not fully address E2E latency-bounded routing across multiple switches under MW scheduling. Existing QoS routing and traffic engineering approaches typically model switches as a set of links with residual capacity, thereby neglecting contention and input–output matching constraints. Deterministic network calculus provides end-to-end bounds from per-hop abstractions but does not directly translate a probabilistic latency requirement into implementable per-flow queue weights under MW scheduling.

III. System Model

Consider a discrete-time packet network, modeled as a directed graph $G = (V, E)$ of routers, where each router is an MW IQ switch. Each vertex $v \in V$ represents a router with n physical ports, where each port consists of one input and one output. Time is slotted with slot length τ (in seconds). Each edge represents a network link, and the maximum packet forwarding rate μ_O (in packets/second) over any link is $1/\tau$, i.e., at most one packet can be transmitted over a link per slot. Traffic is injected into the network as a Poisson process with mean rate λ (in packets/second). Hosts attached to routers act as sources and sinks of flows. For any host access link, the aggregate mean arrival rate of all flows injected on that link satisfies $\sum_f \lambda_f \leq \mu_O$. Traffic is not shaped anywhere.

1) *IQ Router Model*: Each router supports a per-flow VOQ design, as shown in Fig. 1. Specifically, each router is a bipartite

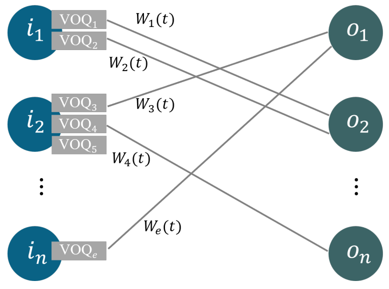

Fig. 1: VOQ multigraph with schedule weights

multigraph, where VOQ_e corresponds to an edge between a physical input $i(e)$ and a physical output $o(e)$. At any given router, each flow is mapped to a unique VOQ. If multiple VOQs share the same input–output pair, they form parallel edges in the multigraph.

In each slot, the router builds a schedule that determines VOQ service with a service decision for all VOQs, denoted by $S_e(t) \in \{0, 1\}, \forall e$. If $S_e(t) = 1$, one packet is served from VOQ_e in slot t. The set of $S_e(t), \forall e$ must satisfy: (i) at most one scheduled VOQ per physical input, (ii) at most one scheduled VOQ per physical output, and (iii) at most K scheduled VOQs in total, i.e., a cardinality constraint on the schedule size. Departures are $D_e(t) = S_e(t)\mathbf{1}\{Q_e(t) > 0\}$, where $Q_e(t)$ is the queue length of VOQ_e at the beginning of the slot. The router runs an MW matching every slot, with per-VOQ weights

$$W_e(t) = \big(\gamma_e Q_e(t) + b_e\big)\, \mathbf{1}\{Q_e(t) > 0\}, \gamma_e \in \mathbb{Z}_{>0}, b_e \in \mathbb{Z}_{\geq 0}. \quad (1)$$

In MW, $S_e(t), \forall e$ are chosen to maximize $\sum_e W_e(t) S_e(t)$ subject to incident-edge feasibility. Empty queues contribute zero weight when building the schedule. As shown in Fig. 1, VOQ_3 and VOQ_4 contend for the same input, VOQ_3 and VOQ_e contend for the same output, and VOQ_1 and VOQ_2 contend for both. A feasible schedule must resolve contention among VOQs on both the input and the output sides.

2) *Flow Requests*: A new latency-sensitive flow request is denoted by $\text{Req} = (s, d, \lambda, T, \varepsilon)$, with $s, d, \lambda, T,$ and ε representing the source, destination, mean packet arrival rate, E2E latency bound, and maximum allowable latency violation probability, respectively. A flow violates its SLA (T, ε) if the percentage of packets whose E2E sojourn time exceeds T is larger than ε. Let $a \triangleq \lambda\tau$ be the mean arrival rate (in packets/slot), and $\widehat{T} \triangleq \lceil T/\tau \rceil$ be the deadline measured in slots. For each flow request, an admission decision must be made to either admit or reject it. If admitted, the decision must also return a router sequence from s to d along with a per-hop reserved service share $q^\star$ conditioned on backlog, defined as a lower bound on $\mathbb{P}\{S_e(t) = 1 \mid Q_e(t) > 0\}$ for the flow's VOQ at each hop. An admitted flow is mapped to a unique VOQ at each router along its path. If the flow is mapped to VOQ_e at a router, $q^\star$ is used to find the corresponding queue-weights (γ_e, b_e).

IV. Per-Flow Queue Model and Bound Inversion

A per-flow VOQ takes one of two classes depending on its proximity to the traffic source. An *edge* VOQ directly observes

Poisson injection, whereas a *transit* VOQ observes packets emitted by upstream scheduling, as such the arrival process is no longer Poisson. For both classes, we model the per-slot service seen by the VOQ using i.i.d. Bernoulli(q) service opportunities, where q is the probability that a backlogged VOQ is offered one service opportunity in a slot.

Let $Q(t)$ denote the VOQ's queue length at the start of slot t, and let $A(t)$ be the number of packet arrivals at the end of slot t. In each slot, let $B(t) \in \{0,1\}$ indicate whether this VOQ is offered service. $\{B(t)\}$ is i.i.d. Bernoulli(q) and independent of arrivals, so a VOQ is served in a slot with probability q (unlike the actual MW schedule indicator denoted $S_e(t)$, which is state dependent and is not i.i.d.). Departures are $D(t) = B(t)\mathbf{1}\{Q(t) > 0\}$, and thus $\mathbb{P}\{D(t) = 1 \mid Q(t) > 0\} = q$.

The queue recursion is

$$Q(t+1) = \big(Q(t) - D(t) + A(t)\big)^+. \tag{2}$$

Two hop classes are used:

- **Edge hop (hop 1):** Exogenous arrivals are Poisson(λ) in continuous time. Aggregated into slots of duration τ, $A(t) \sim \mathrm{Poisson}(a)$ with $a \triangleq \lambda\tau = \mathbb{E}[A(t)]$. A Palm correction is required when analyzing the sojourn time of a randomly sampled packet via size-biasing the arrival batch [16].
- **Transit hops:** A per-flow upstream scheduler can emit at most one packet per slot, implying a lattice arrival model $A(t) \in \{0,1\}$ with $\mathbb{P}\{A(t) = 1\} = p$. We match the mean arrival rate setting $p = \mathbb{E}[A(t)] = a$; since $A(t) \in \{0,1\}$ in this lattice model, this implies $a \in [0,1]$.

Stability requires $\mathbb{E}[A(t)] < \mathbb{P}\{D(t) = 1 \mid Q(t) > 0\} = q$

A. Stationary PGF, Sojourn-time Tail and MGF

For the slotted queue in (2) with service opportunity at slot start and arrivals at slot end, the stationary start-of-slot queue length PGF, denoted by $G_Q(z) = \mathbb{E}[z^Q]$, has a closed form:

$$G_Q(z) = (1-\rho)\,\frac{G_A(z)\,q\,(z-1)}{z - G_A(z)\big((1-q)z + q\big)}, \tag{3}$$

where $G_A(z) = \mathbb{E}[z^A]$ and $\rho \triangleq \mathbb{E}[A]/q$ ($\rho < 1$ for stability).

The post-service queue length $Q^+ = Q - D$ after the slot's service opportunity, before the slot's arrivals, has PGF $G_{Q^+}(z) = G_Q(z)/G_A(z)$. Let X denote the backlog ahead of a tagged packet upon its arrival to the queue. Conditioned on $X = x$, the packet's sojourn time is the number of Bernoulli(q) trials needed to obtain $x+1$ successes:

$$\mathbb{P}\{W > \widehat{T} \mid X = x\} = \mathbb{P}\{\mathrm{Binomial}(t,q) \le x\}. \tag{4}$$

Averaging over X yields the model-exact tail $\mathbb{P}\{W > \widehat{T}\}$. For Chernoff-style composition, the MGF is used. Define

$$g(\theta;q) \triangleq \frac{qe^\theta}{1-(1-q)e^\theta}, \qquad \theta < -\log(1-q). \tag{5}$$

Then

$$M_W(\theta) \triangleq \mathbb{E}[e^{\theta W}] = \mathbb{E}\big[g(\theta;q)^{X+1}\big] = g(\theta;q)\,G_X\big(g(\theta;q)\big), \tag{6}$$

where G_X is the PGF of X. For transit hops, $X = Q^+$. For hop 1, a Poisson arrival model can include multiple arrivals in the same slot. Therefore, G_X includes the Palm correction factor, $\big(1-G_A(z)\big)/\big((1-z)\mathbb{E}[A]\big)$, for the multiple same-slot arrivals that a packet sees. This is more challenging than the case of a periodic constant-bit-rate (CBR) arrival model, where the Palm correction vanishes (i.e., $X = Q^+$), since at most one arrival per slot may occur (i.e., $A \in \{0,1\}$).

B. E2E-Latency Tail Bound via Chernoff–Hölder and Inversion

Consider a path of L hops with per-hop sojourn times $W_1, \dots, W_L$. Let $W_{\mathrm{e2e}} \triangleq \sum_{h=1}^{L} W_h$ denote the end-to-end sojourn time. We seek an upper bound on $\mathbb{P}\{W_{\mathrm{e2e}} > \widehat{T}\}$. By the exponential Markov inequality, for any $\theta > 0$ such that $\mathbb{E}[\exp(\theta W_{\mathrm{e2e}})] < \infty$, we have

$$\mathbb{P}\Big\{W_{\mathrm{e2e}} > \widehat{T}\Big\} = \mathbb{P}\Big\{e^{\theta W_{\mathrm{e2e}}} > e^{\theta\widehat{T}}\Big\} \le e^{-\theta\widehat{T}}\,\mathbb{E}\Big[e^{\theta\sum_{h=1}^{L} W_h}\Big]. \tag{7}$$

Optimizing over θ to tighten the above bound yields

$$\mathbb{P}\Big\{\sum_{h=1}^{L} W_h > \widehat{T}\Big\} \le \inf_{\theta>0} e^{-\theta\widehat{T}}\,\mathbb{E}\Big[e^{\theta\sum_{h=1}^{L} W_h}\Big]. \tag{8}$$

Dependencies between hop delays can arise because a flow's arrivals at a downstream queue are shaped by upstream scheduling. To obtain a tractable upper bound without per-hop budget splitting, Hölder's inequality is used to derive an upper bound on the joint MGF without assuming independence across hops. With exponents $p_h > 1$ such that $\sum_h 1/p_h = 1$,

$$\mathbb{E}\Big[\prod_{h=1}^{L} e^{\theta W_h}\Big] \le \prod_{h=1}^{L} \mathbb{E}\big[e^{p_h\theta W_h}\big]^{1/p_h}. \tag{9}$$

The choice $p_h = H$ for a hop-count budget $H \ge L$ is used in implementation; when $L < H$, dummy factors are used for padding such that $\sum_h 1/p_h = 1$. This yields an explicit bound:

$$\begin{aligned}\mathbb{P}\{W_{\mathrm{e2e}} > \widehat{T}\} \le \inf_{\theta\in\mathcal{D}(H,q)} \exp\Big(&-\theta\widehat{T} + \tfrac{1}{H}\psi_1(H\theta;q)\\ &+ \tfrac{L-1}{H}\psi_{\mathrm{tr}}(H\theta;q)\Big),\end{aligned} \tag{10}$$

where ψ_1 and ψ_{tr} are hop-1 and transit log-MGFs under the imposed models, which are finite when $H\theta < -\log(1-q)$. Here $\mathcal{D}(H,q)$ is the set of θ such that $\theta > 0$, $\psi_1(H\theta;q) < \infty$, and $\psi_{\mathrm{tr}}(H\theta;q) < \infty$.

C. Bound Inversion for q

Fix a hop budget H and an SLA (T,ε). For a given q, define $\mathcal{U}_{H,L}(q)$ as the right-hand side of (10) for a route of length L. Since $\psi_{\mathrm{tr}}(\cdot;q) \ge 0$ for $\theta > 0$, $\mathcal{U}_{H,L}(q)$ is nondecreasing in L, and the worst case over $1 \le L \le H$ occurs at $L = H$. We therefore define

$$q_{\mathrm{req}}(H) \triangleq \inf\Big\{q \in (0,1) : \mathcal{U}_{H,H}(q) \le \varepsilon\Big\}. \tag{11}$$

The monotonicity of the bound in q enables 1D bisection. The monotonicity in H allows early termination once $q_{\mathrm{req}}(H) \ge 1$. Throughout the paper, q and q_{eff} denote conditional service shares (probabilities of being scheduled given backlog).

V. Translation Layer: Share-to-Weight Mapping and Repair

The translation layer of the proposed approach turns a required service share q_{req} into MW parameters (γ, b) that guarantee an effective service share $q_{\mathrm{eff}} \ge q_{\mathrm{req}}$.

A. Effective Service Probability

For $\mathrm{VOQ_e}$, define the one-slot effective service probability conditioned on backlog:

$$q_{\text{eff}}(e) \triangleq \mathbb{P}\{S_e(t) = 1 \mid Q_e(t) > 0\}. \tag{12}$$

Under stability in steady state, rate conservation (i.e, $\mathbb{E}[Q(t+1)-Q(t)] = 0$) implies $\mathbb{E}[D_e] = \mathbb{E}[A_e]$. Let $a_e \triangleq \mathbb{E}[A_e]$. Then, $\mathbb{E}[D_e] = a_e$ and

$$q_{\text{eff}}(e) = \frac{a_e}{\mathbb{P}\{Q_e > 0\}}. \tag{13}$$

Thus, upper-bounding the busy probability $\mathbb{P}\{Q_e > 0\}$ yields a lower bound on q_{eff}.

B. Forced Service

Let $\Theta_e \triangleq \gamma_e + b_e$ be the smallest value that the scheduling weight of $\mathrm{VOQ_e}$ can take when it is backlogged ($Q_e(t) > 0$). For slot t, define the random blocking threshold weight $Z_e(t)$ as the largest scheduling weight among competitor VOQs that can exclude $\mathrm{VOQ_e}$ from service in that slot under the schedule matching constraints (i.e., $S_e(t) = 1$ for only one VOQ): *(a)* among parallel VOQs on the same input–output pair, *(b)* among VOQs that share the same input as $\mathrm{VOQ_e}$, *(c)* among VOQs that share the same output as $\mathrm{VOQ_e}$, and *(d)* the K-th largest weight among VOQs that share neither endpoint with VOQ_e, if K would upper bound the size of a schedule already satisfying *(a)*, *(b)*, *(c)*. If $Q_e(t) > 0$ and $W_e(t) > Z_e(t)$, then every MW schedule must include e, so $S_e(t) = 1$. The key observation is that a backlogged VOQ is guaranteed service in an MW schedule if its own weight $W_e(t)$ exceeds a threshold determined by the heaviest VOQs that can exclude it from service in the current slot.

Let $\delta_e(\eta) \triangleq \mathbb{P}\{Z_e(t) \geq \eta\}$ denote the probability that the blocking threshold is at least η, where η is a scheduling-weight level that can exclude VOQ_e from service.

Then

$$\mathbb{P}\{Q_e > 0\} \leq a_e + \delta_e(\Theta_e), \tag{14}$$

and using (14) to upper bound $\mathbb{P}\{Q_e > 0\}$ in (13) yields

$$q_{\text{eff}}(e) \geq q_{\text{lb}}(e) \triangleq \frac{a_e}{a_e + \delta_e(\Theta_e)}. \tag{15}$$

To make (15) computable without modeling the joint distribution of competing VOQ weights, we upper bound $\delta_e(\eta)$ (and in particular $\delta_e(\Theta_e)$) using only per-VOQ marginal steady-state statistics.

For any competitor VOQ_j, let $W_j(t) = \gamma_j Q_j(t) + b_j$ and define $\pi_j(\eta) \triangleq \mathbb{P}\{W_j(t) \geq \eta\}$, the steady-state probability that VOQ_j's weight is at least η. We then construct a dependence-free bound $\hat{\delta}_e(\eta)$ via a disjoint endpoint partition with a best split that does not assume independence across competitors. VOQs always contend through the schedule constraints *(a)*, *(b)*, *(c)*, when constraint *(d)* causes contention, we upper-bound the probability that at least K non-incident competitors (i.e., competitors that share neither input nor output with VOQ_e) have scheduling weight at least η, using Markov's inequality applied to the exceedance count. The marginal probabilities $\pi_j(\eta) = \mathbb{P}\{W_j(t) \geq \eta\}$ are computed from the per-hop queue models given parameters $(a_j, q_j, \gamma_j, b_j)$.

C. Mapping Condition

A sufficient condition to guarantee $q_{\text{eff}}(e) \geq q_{\text{req}}(e)$ is

$$\hat{\delta}_e(\Theta_e) \leq a_e \frac{1 - q_{\text{req}}(e)}{q_{\text{req}}(e)}. \tag{16}$$

We treat γ_e as a given design parameter; the algorithm then chooses the smallest integer threshold Θ_e satisfying (16) and sets $b_e = \Theta_e - \gamma_e$.

D. Bounded Monotone Repair and Non-stomping Guarantee

Provisioning a new VOQ at a router can change the competitor sets of multiple VOQs at that router, which can invalidate previously provisioned service shares. To prevent "stomping," we run a router-local repair procedure before committing any weights. The repair procedure recomputes $\hat{\delta}_e(\eta)$, our bound on the probability $\mathbb{P}\{Z_e(t) \geq \eta\}$, using the updated competitor sets induced by the additional VOQ.

Given a candidate admission, we recompute the affected per-VOQ probability bounds $\hat{\delta}_j(\cdot)$ and re-check the corresponding feasibility condition (16) for each affected VOQ_j. Any VOQ that violates (16) is repaired by increasing its offset b_j (hence Θ_j), which decreases $\hat{\delta}_j(\Theta_j)$ and enforces its forced-service guarantee. We repeat until all VOQs satisfy (16) or some Θ_j exceeds a configured ceiling $\Theta_{\max}$. Because offsets are integers and only increase, the procedure terminates in finite time. If the repair succeeds, all certified VOQs at the router satisfy $q_{\text{eff}} \geq q_{\text{req}}$; otherwise, the hop is declared infeasible for this admission attempt.

VI. Routing/Provisioning Layer: Turn-Graph Search Oracle

The per-flow VOQ used at router v depends on both the incoming port dictated by the previous hop and the outgoing port to the next hop. Thus, the algorithms path-search must remember the previous router. A *turn* at router v is defined as a triple (u, v, w), where router u is the previous router and w is the next router, i.e., arriving at v from u and departing to w. A turn (u, v, w) is valid if $(v, w) \in E$ and either $(u, v) \in E$ or $u =$ INJECT for host-router links. We form a directed graph G_H whose nodes are arrival states (u, v), and whose directed edges are valid turns: an edge $(u, v) \to (v, w)$ exists iff (u, v, w) is a valid turn. The computed route is not just a sequence of routers over $G = (V, E)$ but a sequence of turns in the turn-graph G_H.

A. Local Feasibility and Cost Check

For a candidate turn (u, v, w) (i.e., placing the new flow into a candidate per-flow VOQ_e at router v) and a required service share q_{req}, the routing algorithm calls a router-local feasibility oracle:

$$(f, \Delta R, \text{Patch}) \leftarrow \text{LocalFeasibleAndCost}(u, v, w; q_{\text{req}}). \tag{17}$$

Here $f \in \{0, 1\}$ is a feasibility flag: $f = 1$ iff the oracle finds a weight-update patch that makes the turn locally feasible. The oracle checks whether the new VOQ_e can be given at least q_{req} service share at router v. The oracle does this by determining whether there exists an integer weight floor $\Theta_e = \gamma_e + b_e$ such that $q_{\text{eff}}(e) \geq q_{\text{req}}$. It runs a monotone search such that the new VOQ$_e$ satisfies $q_{\text{eff}}(e) \geq q_{\text{req},e}$ and all existing competitor VOQ$_j \forall j$ satisfy $q_{\text{eff}}(j) \geq q_{\text{req},j}$. The oracle returns: (i) $f \in \{0,1\}$; (ii) an integer resource increment $\Delta R \triangleq \sum_{j \in \text{Patch}} \Delta b_j$, defined as the total increase in queue-weight offsets in a commit-ready patch. The algorithm is restricted to simple paths with no repeated routers such that per-turn weight mappings do not interact, so as not to break the monotonicity.

B. Least-cost Routing Via Branch-and-bound

Given a hop budget H, the inversion of the E2E latency bound provides a required per-hop share $q_{\text{req}}(H)$. We construct a pruned turn graph G_H from the physical topology by retaining only the turns (u, v, w) that are locally feasible under $q_{\text{req}}(H)$. An edge in $G_H : (u, v) \rightarrow (v, w)$ is assigned an additive cost $\Delta R(u, v, w)$. The algorithm returns a turn-graph path from the start node (INJECT, s) to any destination node $(\cdot, d)$ that minimizes the total cost along the path. We solve this optimally using branch-and-bound. For a path prefix P ending at node (u, v), let $g(P)$ denote the cost accumulated so far. As an admissible lower bound LB on the remaining cost, we precompute the relaxed shortest-path distance $d_{\text{relax}}(u, v) = \sum \Delta R$, defined as the minimum remaining cost from node (u, v) to any destination node when the simple-path constraint is ignored. Since dropping constraints cannot increase the optimal cost, a valid LB on the best completion of P is:

$$\text{LB}(P) \triangleq g(P) + d_{\text{relax}}(u, v). \tag{18}$$

The search maintains an upper bound UB, defined as the cost of the best complete path found so far, and prunes any prefix P with $\text{LB}(P) \geq \text{UB}$.

If Algorithm 1 returns "Accept", each hop provides at least $q_{\text{req}}(H)$, meaning each hop is locally feasible after the repair procedure in section V.D, such that all VOQs satisfy $q_{\text{eff}} \geq q_{\text{req}}$. The path has minimum cost in $\sum \Delta R$, and is found via branch-and-bound. Algorithm runtime is discussed in section VII.B.

Algorithm 1 Tail-feasible admission and least-resource routing

```
Require: Graph $G = (V, E)$, request $(s, d, \lambda, T, \varepsilon)$, slot $\tau$,
    max threshold $\Theta_{\max}$
Ensure: Reject, or Accept with route and weight atomic patch
 1: $a \leftarrow \lambda\tau$; $\widehat{T} \leftarrow \lceil T/\tau \rceil$
 2: Best $\leftarrow \emptyset$
 3: $H_{\min} \leftarrow$ ShortestHopCount$(G, s, d)$
 4: for $H = H_{\min}$ to $|V| - 1$ do
 5:     $q_{\text{req}} \leftarrow$ InvertExactE2E$(\widehat{T}, \varepsilon, H, a)$
 6:     if $q_{\text{req}} \geq 1$ then
 7:         break
 8:     end if
 9:     Build $G_H$ by evaluating LocalFeasibleAndCost on all
    turns and retaining $f = 1$ turns with cost $\Delta R(u, v, w)$
10:     Compute relaxed shortest-path distances $d_{\text{relax}}$ in $G_H$
11:     $(P, \text{cost}, \text{Patch}) \leftarrow$ BranchBoundLoopFree
    $(G_H, d_{\text{relax}}, s, d, H)$
12:     if $P \neq \emptyset$ and (Best $= \emptyset$ or cost $<$ Best.cost) then
13:         Best $\leftarrow (P, \text{cost}, \text{Patch}, q_{\text{req}}, H)$
14:         if Best.cost $= 0$ then
15:             break
16:         end if
17:     end if
18: end for
19: if Best $= \emptyset$ then
20:     return Reject
21: end if
22: Commit Best.Patch
23: return Accept, Best.P and $q^{\star} =$ Best.$q_{\text{req}}$
```

VII. Simulation Results

We compare the proposed admission and routing algorithm against three baselines on the same discrete-time network simulator. The baselines are: B1 (MIRA-family minimum-interference routing), B2 (LARAC-style constrained shortest path), and B3 (deterministic-network-calculus bottleneck sweep). Since the baselines cannot break a total latency target into per-hop service shares or map the service share to queue weights, all baselines invoke the same mapping functions used by our algorithm, and run over the same turn graph G_H.

A. Simulation Settings

Experiments are conducted for 25- and 100-router random network topologies. An "edge router" is generated with probability 0.3 which has $[3, 6]$ attached hosts, while all other routers (0.7) are "core routers" that have 1 attached host. Routers have an average degree of 2.5 in the 25-router experiment, and 5 in the 100-router experiment. The service rate is $\mu_O = 100$ packets/s ($\tau = 10$ ms) with a schedule size of $k = 10$. For each flow: $\lambda \in [0.12\mu_O, 0.20\mu_O]$ packets/s, $T \in [50, 500]$ ms, and $\varepsilon \in \{10^{-2}, 10^{-3}\}$ are all uniformly sampled. 100 flows arrive sequentially every 30 slots during the flow arrival window ($\approx$ 30s), after which we measure a steady state window for 2000 slots ($\approx$ 20 s).

B. Performance Results

Fig. 2 shows SLA violations in the 25- and 100-router settings. In both cases, all baselines persistently violate the latency bounds even in steady state $[t = 30, t = 50]$s, while the proposed algorithm has zero violations in both the admission window $[t = 0, t = 30)$s and steady-state measurement window. In the 100-router case, the violation rate of the baselines improves, due to the greatly increased network capacity under the same offered load, but remains nontrivial.

Fig. 3 shows the admitted load in the 25-router and 100-router settings. In the 25-router case, the proposed algorithm admits about 33% of the admitted load by the baselines. In the 100-router case, this gap reduces as the proposed algorithm admits 44% of the admitted load by the baselines. All runtimes were measured on a machine with a 5 GHz AMD 9800X3D CPU. The proposed algorithm's mean runtime which resulted in "Accept" was 2352 ms with p95 of 23545 ms for the 25-router case, and 71223 ms with p95 of 557123 ms for the 100-router case. The branch-and-bound finished in sub-ms time (0.0066 ms, 0.05 ms). Total runtime was dominated by an unoptimized implementation of the polynomial-time feasibility and monotone search oracle, accounting for over 99% of the

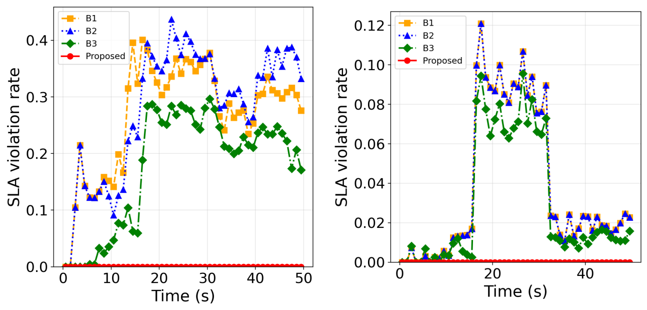

Fig. 2: SLA violation rate vs. time (L: 25-router; R: 100-router).

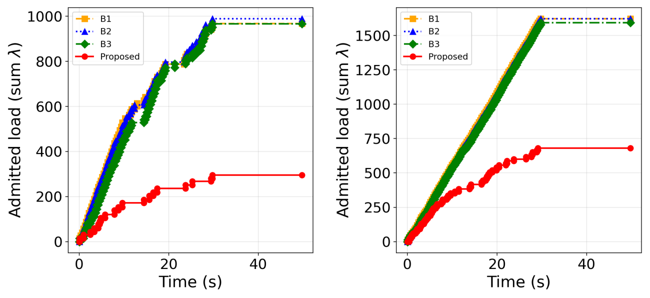

Fig. 3: Admitted load vs. time (L: 25-router; R: 100-router).

runtime. The implementation is amenable to optimizations such as memoization, precomputation, and reuse of per-router aggregates.

The key observation is that there exists a viable admitted load under a specific flow routing configuration that yields zero violations. In Fig. 3, B3 reaches a near-zero violation rate at $\approx$ 800 packets/s admitted at $t = 17$ in the 100-router case. This is only $\approx$ 100 packets/s ($\approx 12\%$) above the proposed algorithm, suggesting that the maximum admissible load for zero violations is not substantially higher than what is admitted by the proposed algorithm. This highlights a clear tradeoff: our algorithm admits less load but maintains SLA compliance for all admitted flows, whereas the baselines over-admit traffic and incur immediate and sustained SLA violations. The proposed algorithm performs a feasibility check, admitting a flow only if its selected route is feasible and ensures the SLA guarantees of previously admitted flows. This "non-stomping" comes at marginally increased conservatism during admission. This conservatism is largely due to the slack between the model and physical MW switch rather than the closed-forms given by the model itself. E2E Chernoff-Hölder avoids assuming independence between flows, but yields a worst-case hop-budget $q_{\text{req}}(H)$ which is applied uniformly at each hop, creating global slack while only some hops are tight. Building the dependence free bound $\hat{\delta}_e(\eta)$, as well as the Markov bound on the schedule size K exceedance counts can overestimate how backlogged a VOQ can be at each hop. This backlog overestimation, and the worst-case $q_{\text{req}}(H)$ yield larger queue weight Θ values. The larger required Θs result in a mutual race to $\Theta_{\max}$ among competitor VOQs, invalidating the hop for the admission attempt. In contrast, the baselines are far more optimistic in admission, using heuristic criteria to admit and route, which do not prevent already-admitted flows from being pushed beyond their latency bounds.

VIII. Conclusion

We propose a new approach for joint flow admission and routing with SLA compliance under MW scheduling. The key idea is to decompose the E2E latency bound into a tunable per-hop service-share parameter, which is mapped to queue weights and incorporated into routing decisions to enforce the E2E latency constraint. During the admission window when the network is under persistent perturbations, the proposed algorithm maintains admitted flows' SLAs, demonstrating robustness to dynamic edge-network conditions. This SLA compliance comes at the cost of reduced admitted load; however, the admitted-load gap relative to the best baseline in the near-zero-violation regime is modest. A potential future direction is to explore a Markov-modulated service model to better capture bursts and service coupling.

References

[1] Z. Xu, X. Chen and Z. Zhu, "INT-assisted adaptive packet scheduling in PDP switches for end-to-end latency control," *IEEE INFOCOM*, London, United Kingdom, 2025, pp. 1-10.

[2] ETSI, "Multi-access edge computing (MEC); Use cases and requirements," ETSI GS MEC 002 V4.1.1, Jun. 2025.

[3] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Trans. Autom. Control*, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[4] A. L. Stolyar, "MaxWeight scheduling in a generalized switch: State space collapse and workload minimization in heavy traffic," *Ann. Appl. Probab.*, vol. 14, no. 1, pp. 1–53, 2004.

[5] J. Nair, A. Wierman, and B. Zwart, "When heavy-tailed and light-tailed flows compete: the response of delay stability to parameter changes," *IEEE/ACM Trans. Netw.*, vol. 24, no. 4, pp. 2056–2069, Aug. 2016.

[6] P. Van Mieghem and F. A. Kuipers, "On the complexity of QoS routing," *Comput. Commun.*, vol. 26, no. 4, pp. 376–387, Mar. 2003.

[7] M. Andrews and M. Vojnović, "Scheduling reserved traffic in input-queued switches: new delay bounds via probabilistic techniques," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 595–605, May 2003.

[8] S. Iyer and N. McKeown, "Using constraint sets to achieve delay bounds in CIOQ switches," *IEEE Commun. Lett.*, vol. 7, no. 6, pp. 275–277, Jun. 2003.

[9] M. Ajmone Marsan, P. Giaccone, E. Leonardi, and F. Neri, "On the stability of local scheduling policies in networks of packet switches with input queues," *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 642–655, May 2003.

[10] S. U. Nabar, N. Kumar, M. Bayati, and A. Keshavarzian, "Achieving stability in networks of input-queued switches using a local online scheduling policy," in *IEEE GLOBECOM*,St. Louis, MO, USA, 2005, pp. 694-698.

[11] A. Jüttner, B. Szviatovszki, I. Mécs, and Z. Rajkó, "Lagrange relaxation based method for the QoS routing problem," in *IEEE INFOCOM*, vol. 2, Anchorage, AK, USA, 2001, pp. 859–868.

[12] M. Kodialam and T. V. Lakshman, "Minimum interference routing with applications to MPLS traffic engineering," in *IEEE INFOCOM*, Tel Aviv, Israel, 2000, pp. 884–893.

[13] S. Bondorf, P. Nikolaus, and J. B. Schmitt, "Quality and cost of deterministic network calculus: design and evaluation of an accurate and fast analysis," *ACM Meas. Anal. Comput. Syst.*, vol. 1, no. 1, Jun. 2017.

[14] M. Ciucu and J. B. Schmitt, "Perspectives on network calculus: no free lunch but still good value," in *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, pp. 1–25, 2014.

[15] J.-Y. Le Boudec, "Applicability of network calculus to DetNet," in *IETF 100 DetNet Working Group Meeting*, Nov. 2017.

[16] K. Sigman and W. Whitt, "Marked point processes in discrete time," *Queueing Systems*, vol. 92, no. 1–2, pp. 47–81, 2019.